YACAS - A Batch Computer Animation System

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master of Science in Computer Science
in the
University of Canterbury

by
Thomas J. Britton

University of Canterbury
1978

Abstract

This thesis describes the design and implementation of a computer animation system called YACAS. YACAS is a batch animation system consisting of two parts. One part is a set of subroutines written in Burroughs Extended ALGOL that provide a number of functions for use by an animator in preparing a program to make an animated film. The second part is a program which the animator can use to interactively display and record his film.

Pictures produced by the system are 2 dimensional "wire-frame" images in black and white. In YACAS the data structure recognizes a distinction between the shape of a picture (referred to as a "cel") and other picture attributes (position, size, etc). Pictures may be "simple" or may be joined and manipulated as a hierarchical group called an "articulated" picture. A small number of commands have been provided to allow motions of pictures and the display window to be described. A mechanism is provided to allow the user to add new motion commands. An arbitrary number of motions may be flexibly combined to produce concurrent asynchronous motions. A compact form of film file is produced by the user's animation program which can be displayed with the interactive playback program.

The main body of the thesis describes YACAS as it has been designed and implemented. The last chapter of the thesis describes a number of enhancements that would make YACAS more versatile. Appendix A is a complete User Guide for the system, while Appendix B gives an example of the programming and use of the YACAS subroutines.

Acknowledgements

I would like to thank professors Les Mezei and Ron Baecker of the Computer Systems Research Group, University of Toronto, with whom I worked for several years before beginning this thesis. They provided my initial introduction to the enjoyable functions of computers - particularly their uses in art and animation - and gave me continuing support, encouragement, and inspiration.

CBL (Canterbury) Ltd. has been most generous in allowing me access to their timesharing computer system and text formatting program which has been used in the preparation of this thesis.

I thank my thesis supervisor, Dr. M. Maclean, for his encouragement and many helpful comments.

Finally, I am indebted to my wife, Noreen, for her help, encouragement, and understanding, particularly in these last trying weeks.

Contents

Chapter I

Introduction

I.1 Animation

Animation is a group of techniques used in the production of films. Animation may be characterized by two factors which make it different from "live-action" (conventional) film-making: (a) the subjects filmed are usually inanimate (clay models, paper cut-outs, coloured drawings), and (b) the film is usually produced on a frame-by-frame basis (a composite of objects is assembled, photographed, altered slightly, photographed again ...). The familiar form of animation - cartoons (Bugs Bunny, Roadrunner, Mickey Mouse, etc.) - is only one application; another very broad category could consist of films that explain things: specific teaching films, explanatory films (how something works), advertising commercials.

Any number of methods have been and may be used to produce animated films. Drawn images may be used: drawn on paper, clear plastic (Cel animation - see below), chalk boards, or even directly on the raw film stock. Solid objects may be used: the puppets of Jiri Trnka, clay models, potatoes (see the Nov. 1977 Filmmakers Newsletter), even people (for example: Norman McLaren's "Neighbours" (1952)). Other techniques include the Alexeieff "pin-board" (a square board with 250,000 pegs that may be pushed in and out of its surface; pictures are formed by pushing the pegs in to varying heights and lighting the board obliquely - the shadows combine to form the images). Cut-out animation is accomplished by cutting out bits of coloured paper, photographs, etc and forming pictures as collages; the individual bits may be moved to form successive images.

Sand has been used by forming images with piles of sand on a translucent glass plate and lighting it from below. In short, anything that can be used to form a series of pictures may be used to produce an animated film.

Of the various techniques available, three are most commonly used. Object animation is frequently used in advertising: boxes of soap powder marching about the screen. Cut-out animation is relatively quick and easy to produce, and has been used extensively in the Monty Python television series for example. By far the most-used technique though, is cel animation. Cel animation is used for everything from the low-quality television series such as "The Flintstones", through title and credit sequences for conventional films (the title sequences for the "Pink Panther" films are delightful), to full-length feature animated films such as Walt Disney's "Fantasia" (1941) or Ralph Bakshi's "Fritz the Cat" (1972).

Cel animation will be discussed in more detail below as it forms a basis for understanding the form of animation produced by YACAS.

Readers interested in learning more about animation will find several references in the Bibliography. "The Technique of Film Animation" [Halas:1971] is one of the most thorough books, while "The Do-it-yourself Animation Book" [Godfrey:1976] is an excellent introduction to the topic. "Art in Movement" [Halas:1970] discusses a wide range of animation techniques superficially, and shows a great many example pictures.

I.1.1 Cel Animation

The process of cel animation is superficially very simple: images are drawn on clear acetate "cels"; these are placed one on top of the other to produce a composite picture, which is then photographed with a high-precision

camera. Motions are produced by moving individual cels within the group or by replacing individual cels. A specially-designed "table" (an animation-stand) can be used to aid this process. The process can be very time-consuming and tedious owing to the large number of cels to be produced and kept track of.

The Storyboard

A conventional "live-action" film is usually planned on the basis of a script; an animated film is more often planned on the basis of a "storyboard". The storyboard is a graphical outline of the film resembling the familiar comic strip, but usually also includes some indication of how to get from one frame to the next.

Other lists of information may be prepared as part of the planning and organizing stages. One such list is the "dope sheet". This is used to record all the information for a scene: an outline of the action, the sound effects and script, the specific cels to be used and the camera instructions for each frame.

Cels

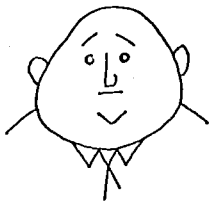
A composite picture usually consists of a "background" containing elements of the picture that do not change and other cels that may be changed. Motions in an animated film are usually accomplished with one or more of three principle techniques.

(1) Individual cels may be moved or rotated relative to the others (or the background);

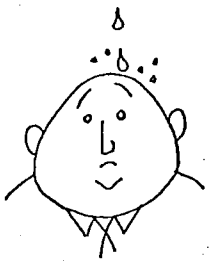
(2) Individual cels may be replaced with others (the mouth motions of a character talking are usually accomplished by having several cels of the mouth in different shapes and exchanging the cels);

(3) by altering the position/orientation of the camera ("pans" by horizontal movements, "zooms" by vertical

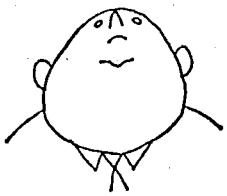
movements, and "spins" by rotations).



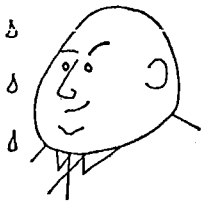
1
man



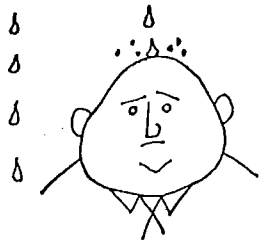
2
man feels water drops on head. Expression changes



3
man looks up. Head shape remains the same but features change



4
man moves sideways to miss the drops. Head seen in three-quarters profile



5
new stream of drops on man's head. Return to 2 with slight change of eyebrows

Fig. 1-1
A simple storyboard
- from [Kinsey:1970] pg. 44

The Animation Stand

The set of cels, to be photographed, must be placed somewhere: a device designed for this purpose is an "animation stand". Simplistically, it consists of a flat table-top, an upright support to hold the camera above and pointing down onto the table sufrace, and usually brackets for lights etc. In practice there is much more complication: for cels to be stacked together to form coherent pictures repeatably they must be precisely placed: an animation stand usually has registration lines and scales ruled on it, and one or more "peg bars" - precisely positioned pegs to fit pre-punched holes in the cels. To aid the process of moving one cel relative to the others, one or more of the peg bars may be able to be moved, usually by means of turning a threaded rod; the amount of movement produced is usually read from an attached dial (the gradations typically correspond to movements in the order of one or two thousandths of an inch). The height of the camera above the stand's surface may also be altered by turning dials (with an associated cam to maintain focus). Occasionally part of the stand's top can be rotated to produce spins; again controlled via dials.

The Process

Large-studio animation can be a massive problem in logistics. An animator develops an idea into a storyboard and makes up design sketches of the characters, backgrounds etc. Assistant animators draw outlines for key frames in the film - on separate cels for each part of a frame that may be moving. Inbetweeners draw the intermediate outlines. Inkers fill in the outlines and add the colours, while checkers ensure that the cels are done properly, that the colours are right, etc. Finally the cameraman and his crew must make use of these cels in the correct order, precisely

positioned, following a carefully prepared dope sheet, to produce the film itself (usually in short scenes which must then be edited together). Even in a small studio these tasks must still be done, though by fewer people. A minute's worth of film (1440 frames) produced in this manner can take from several weeks to a year or more to complete.

I.2 Automating Animation

The above description of cel animation is admittedly biased. Simplifications, standardizations, re-organizations, and a host of other small things can improve the efficiency of animated filmmaking greatly. However, some central limitations remain. The person conceiving the film is often far removed from its actual realization (in time and/or in implementation); the process is inherently time-consuming; there is much room for error; and there is a very heavy reliance on intuition in such matters as how objects move and interact.

Since there are many repetitive aspects to animation, it would appear to be an ideal candidate for computerization.

Computers have been applied to the production of animated films in two significantly different ways. The first - computer control of the animation stand - has become fairly commonplace in the business of animation due to the simplicity and "naturalness" of its use. The second - computer generated graphics - appears to offer greater potential in terms of eliminating the need for the inbetweening, inking, and photographing manual steps, but has not enjoyed equivalent popularity in the business.

I.2.1 Automating the Mechanics

Computers have been, and are currently used to control animation stands [NFB:1970]. A large production animation stand may have a dozen or more precision dials which must be turned just the right amount for each frame's movements. Moreover, the amount that the dials must be turned must be calculated with arithmetic of modest complexity. A mini-computer hooked up to an animation stand through proper interfaces and directed with an appropriate command scheme can do both jobs more quickly and reliably than they can be done by hand.

Another application along this same line was used in the recent movie "Star Wars". Sequences of flying and battling spaceships were shot, naturally enough, using models. The models and their actions were controlled by a computer. This meant that sequences could be repeated with small refinements until they looked right, and actions could be more complex than would be possible by normal manual methods [Pye:1977].

I.2.2 Automating the Graphics

More promising than the electro-mechanical applications mentioned above, is the direct application of the computer's ability to store, manipulate, and draw pictures. Since this is the most time-consuming, error-prone, and labour-intensive part of the animation process, reductions here can be of great value.

Computers can draw pictures, given the proper equipment. Pictures can be drawn on paper using any of several different forms of paper plotters; pictures can be drawn on television-like display screens (graphic displays); or "directly" onto film using graphic Computer Output Microfilm (COM) units. Images may range in complexity from simple "wire-frame" images in black and white, to full-

colour three-dimensional appearing objects. The computer can also manipulate the pictures: they can be re-positioned, their size and/or orientation changed, and even their shape changed. Since the computer can draw individual pictures, it can be made to make movies by drawing a set of pictures, and photographing each separately to produce an animated film.

In the most straight-forward approach, a specific program may be written to make a specific film. A better way is to devise a "system" of programs which can be used to make any desired film (or at least a large set of films). Such a system may make it easier for the user to write a program for a specific film, or may obviate the need for the user to write a program altogether.

I.3 YACAS

This thesis describes the design and implementation of a "system" that can be used to make animated films. This system is one that automates the graphics.

Other computer animation systems have been developed, mostly in Europe and North America. These come in many forms, with a wide range of capabilities. Chapter II discusses the design of several of these systems briefly.

The system that is the subject of this thesis is called YACAS - Yet Another Computer Animation System. This name was chosen because the system is just that - yet another one. It has been developed for two primary reasons.
(1) Computer animation is a useful tool for educators and researchers, and one was not available for use here (the University of Canterbury).
(2) For further research into computer animation to be encouraged, some basic system is necessary.

YACAS is not "new" in the sense that most of its ideas are drawn from earlier systems. Two aspects of YACAS are claimed to be significantly different than other systems, and to offer some useful advantages.

(1) The method of picture organization and storage (see III.3.2 and IV.2.1) combines some ideas from SKETCHPAD [Sutherland:1963] with some from GRASS [DeFanti:1973] to provide a flexible way to store and manipulate complex pictures.

(2) The separation of function between the main film-making program and the film previewing system (see III.3.6 and V.2) results in a more compact film file than is commonly the case, and adds capabilities to the preview program.

Chapter II

Computer Animation Background

This chapter discusses some of the ways that other workers have applied computers to the production of animated films. Other general discussions along similar lines may be found in [Youngblood:1970], [Stephenson:1973], [Halas:1974] and [Wein:1976]; that of Wein is by far the best.

II.1 Computer Graphics - the Basis of Computer Animation

Computer graphics is the field of computer science concerned with the use of computers to store and display pictures. It relies on specialized hardware devices for the means of picture display, and on some sophisticated programming techniques to overcome some of the problems involved. The reader of this thesis is expected to have some understanding of the ideas of computer graphics, and some understanding of the types of hardware available. The book "Principles of Interactive Computer Graphics" [Newman:1974] is an excellent (if lengthy) survey of the field. A shorter introduction may be found in [Dowling:1976]. Some comments, however, are in order to indicate the special needs of computer animation.

Output Devices

Computer graphics is primarily concerned with the production of single images; the time taken to produce each image is of secondary importance to the ability to actually create it. Animation is concerned with the creation of moving pictures produced as a sequence of static ones. The motion is a result of the rate at which these static pictures are displayed (16 pictures per second is usually given as the nominal rate necessary). Where computer-

produced animation is to be displayed directly (as in interactive systems) the graphic display used must be capable of displaying moderately complicated pictures fast enough to be viewed as having motion, and ideally at the standard film rate (24 per second). Slower displays may be used for animation, but the ability to see the motions as they would appear on film, and to be able to control these motions, is hampered.

Another factor of concern to the animator more than to the general graphics user is display stability. Since animation is concerned with the control of motion, instability in the display and the uncontrolled motions so introduced, are highly undesirable. The need for display stability is even greater where the film is to be recorded directly from the display: since the film is usually projected on a screen larger than the display, any instability is magnified.

One particular type of hardware device has been of particular use to the computer animator - the graphic Computer Output Microfilm recorder (COM). These units - which consist of a high-precision display mounted in front of a camera as a single unit - provide the animator with the means of recording animated material with the ease of using a paper plotter. The combined effects of a trend away from graphics capabilities in COM units, and growing availability of video recording and playback equipment (and its use in computer graphics systems) has reduced the importance of COM to the animator.

Input Devices

The ability to prepare pictures for use in programs is as necessary for the animator as it is for the general graphics user. Some of the precision required for computer-aided-design applications is of lesser importance to the animator, but ease of use is perhaps more so. In

interactive systems the user uses the input devices (particularly light-pens and tablets) both to prepare pictures to be animated, and to "show" the computer how the animation is to be done.

One point of departure from common graphics terminology that will be used in this thesis is the use of the expression "gesturing device" to stand for such devices as light-pens, data tablets with their stylus, or any device used to point, sketch, or gesture to the computer. Implied in the use of the expression is the ease with which an animator's hand motions may be used to sketch a picture or to define a motion graphically.

II.2 Computer Animation Systems

The following discussion will describe some of the approaches that have been used in the creation of computer animated films. A number of "systems" have been developed in the past and several of these will be briefly described to show how others have tackled the problem.

II.2.1 The Basic Process

Computer animation is the process of producing sequences of moving images from static pictures using the computer as the main picture drawing and manipulating tool. The process usually involves the following steps.

(1) The film's subject matter is planned. Pictures are designed, decisions made concerning aesthetic aspects such as colour, sound, subject treatment, etc.

(2) A program is written, or an existing animation system is used, to produce the film. The film may be made as a whole, or in parts to be edited together later. Depending on the nature of the system used to make the film, the film material may be stored in files for subsequent recording, previewed on a graphic display before recording, or

interactively created and previewed.
(3) The previewing stages are similar to program debugging runs. The film material is studied and changes are made to correct the logic, coding or aesthetics.
(4) The animated material is recorded. Generally film is used as the recording medium (usually 16mm), though video-tape is sometimes used if available.
(5) In some cases, further steps follow (beyond simple developing of the exposed film). If the film material has been recorded as several separate scenes, these will need to be edited together. If sound is to be added, this step will be necessary. If colour is to be added the appropriate optical process must be used.

The computer's role in the process is to produce the primary sequences of pictures according to the directions specified by the animator. Supplementary functions that may be performed by the computer include picture data encoding and storage, camera control during film recording, and various levels of information storage at intermediate stages. Films may be made by writing programs to do these tasks in a "one-off" manner, and indeed many films have been made in this way. Alternatively, "systems" of programs or subroutines may be produced which do the computer's tasks in a general way so that others may make specific films more easily.

Since this thesis deals with the design of a computer system to aid the production of animated films, the following sections will discuss the way others have tackled the basic problem of creating sequences of pictures. Of particular concern is the way the changes from one frame to the next are described. The film (or video-tape) aspects of the process will not be discussed except as side-issues where particularly relevant to a specific system; some of these issues are discussed in more detail in [Winkless:1968], [Gattis:1970], [Tucker:1972], and

[Burtnyk:1973].

II.2.2 Batch Animation

In essence, a batch animation system consists of a "language" (a specific animation programming language or, more commonly, a set of subroutines to be used within a host language) used by a programmer to prepare a program to make a film. The program is processed in a conventional batch processing environment, usually producing the film information as a file on magnetic tape. The film information is recorded onto film as a separate step. Different computers may be involved in these two steps.

A person making a film using a batch system writes a program. This program describes the pictures to be used (either by using stored, pre-digitized picture descriptions, by using language statements, or, in more primitive systems, by providing arrays containing vector descriptions of the pictures), then the transformations to be performed on these pictures. The descriptions of the transformations include timing information (when operations are to start and end), the names of pictures to be operated on, and parameters controlling the transformations. Usually a large set of transformations is provided by the author of the system from which the user may chose those most appropriate to his needs.

BEFLIX

A number of batch animation systems have been developed. BEFLIX [Knowlton:1964, 1969] was one of the earliest animation systems developed. The "drawing space" for pictures consisted of a grid of numbers (values of 0 to 7). Pictures were created on this grid by setting appropriate patterns of numbers in the grid. For output, the whole grid was used for the final image with the numbers transliterated to blobs of grey of varying shades according

to a table supplied by the user. Commands existed within the BEFLIX language to change the position, size and shape of sub-areas of the grid.

BEFLIX was a good beginning and was used to make several excellent films [Knowlton:1966, Vanderbeek:1967]. The number-grid form of picture representation was well-suited to the creation of richly textured images. Its usefulness was limited, though, because extensive sophisticated knowledge of the host programming language was required to obtain any but the basic movements.

ZAPP

ZAPP [Guerin:1973] is an animation system written as a set of FORTRAN subroutines. ZAPP routines allow the user to describe pictures as sets of vectors in a large drawing space or to save and retrieve picture descriptions from disk storage "libraries". A modest number of commands are available to change the size, position, orientation and shape of pictures, with methods available to control the rate at which the transformations are applied. A versatile method of describing which particular transformations are to be applied to each picture allows an arbitrary number of transformations to different pictures to be carried out simultaneously. ZAPP provides a mechanism whereby users may provide their own transformation commands.

ZAPP's control structure, picture libraries, dynamics routines, and extension capabilities make it an excellent film-making tool. Its main limitations lie in the area of picture representation: no hidden line removal scheme is provided; pictures may consist of coloured vectors but area-shading and vector texturing are not available.

ANTICS

ANTICS [Kitching:1973, 1976] is an animation system also written as a set of FORTRAN subroutines. The capabilities of ANTICS appear to be similar to those of ZAPP, but a much larger set of transformation commands is provided ([Ballam:1976] gives a list of 20 command names with no specific reference to what each does). ANTICS was devised by a designer/animator to be used as a commercial system for animated film-making.

Antics too appears to be an excellent animation tool (insufficient details are available in the literature to access it properly). The wide range of picture transformations listed by BALLAM indicate that skilled animators might be better able to describe their ideas to a programming assistant than might be the case for other systems.

II.2.3 Interactive Animation

One of the less desirable properties of batch animation systems is the time involved in making films. There is the usual delay between submitting the program and getting the results, then the added time required to record or preview the film. When errors are found (as they inevitably are), the process must be repeated to correct them. A second undesirable property is that the batch scheme requires the animator to write a program - an additional skill that some potential film-makers (artists, educators) may find difficult to master. A well-written interactive animation system can surmount these difficulties.

An interactive animation system is usually centered around a graphic display and some form of gesturing device (such as a digitizing tablet with stylus, or a light-pen). The user interacts with the animation system by selecting commands from "menus", sketching pictures with the gesturing

device, and controlling the animation through gestures, sketched pictures, typed information, or other forms of control (function keys, dials, joysticks or whatever is available). The animation system provides the basic set of facilities and the means for the interaction; it is guided by the user's actions and can be made to provide the user with helpful information about the use of the various commands.

One of the chief advantages of an interactive system is the immediate feed-back the user gets concerning the film sequences being developed. The appropriate commands can be entered, the sequence viewed and, if there are changes to be made, the changes can be made immediately and their effects observed. A well-written interactive system allows the user to combine the transformation commands in a highly flexible way so that experimentation is encouraged.

Such a system tends to be more easily learned and used by the non-programming user. Commands do not need to be learned in such detail as in a batch system because their use and effect is quickly made apparent. Films do not need to be as laboriously planned, particularly the timing of motions because experimentation and immediate feedback can be used to choose the best timing. Pictures are more easily prepared for use because they can be "shown" to the system using the gesturing device rather than described as sets of numbers.

Interactive systems are less useful (than batch ones) for the researcher and some forms of educational film. The interactive system can usually not be easily combined with a researcher's simulation program to produce film output; tables of experimental data cannot easily be transformed into dynamic images.

GENESYS

Among the interactive animation systems that have been developed, GENESYS [Baecker:1969, 1974] was the earliest significant system and has served as a model for later ones.

GENESYS used a data tablet with stylus, a graphic display, and a key-board terminal for interaction with a program on the M.I.T Lincoln Laboratory's TX2 computer. Pictures could be sketched, stored and later retrieved for use. Motions were controlled by other pictures called "P-Curves". A P-curve specified both the amount and direction of motion, and the rate at which the motion occurred. P-curves were sketched with the stylus and data points were chosen at a fixed time interval corresponding to the frame rate of the film. The shape of the curve was used to control the amount of motion while the spacing of the data points controlled the rate of the motion. Motions could be easily modified and experimented with by re-drawing or modifying the p-curves.

GENESYS did not provide any algorithmic method to change the shape of a picture. Where a picture's shape had to change a new picture was needed and substituted for the original. The mechanism for determining which pictures were to be used for each frame of a sequence was called a "selection graph". Selection graphs, like pictures, could be saved in the computer for later recall and re-use. Selection graphs, P-curves, and pictures - each independent of the others - could be combined to produce a specific sequence.

GENESYS was a superb animation system (though its life was short - the few years of its development in 1968-1970). Its major limitation was that no method was available to algorithmically produce film sequences. Minor ones were that there was no way to algorithmically change the shape of a picture, and no way for the user to add new commands.

These would have limited its use, but its superb interactive nature would have made it useable by a wide range of people. The demonstration film produced using it [Baecker:1970] was largely made by an animator with no computing experience.

The NRC System

As part of a project to study the way people interact with computers, the National Research Council of Canada (NRC) has developed an interactive animation system (unlike most others, they have attached no name to their system; it is simply referred to as the "NRC system") [Burtnyk:1971].

In the early days of its development, a variety of devices were used to provide users with means of interacting with the computer (a "mouse" (see [Newman:1974], pg.174), various potentiometers, joysticks, and function keys). The current system still retains some of the earlier devices but relies mostly on a stylus and tablet, and a keyboard terminal.

Menus are used as the prime means of selecting operating modes and options. Pictures may be sketched and edited, then stored in picture libraries for later use. The principal method used to create motions in the NRC system, is the ability to change a picture's shape continuously with time (unlike picture editing which can be thought of as a discrete change). The process involves the preparation of two pictures - one having the shape of the picture as it is to appear at the beginning of a sequence, and one having the shape it is to have at the end. The computer algorithmically produces the intermediate pictures using a technique called "linear interpolation" (the technique is described in [Burtnyk:1971] and in the YACAS User Guide (Appendix A) section 2.4).

The NRC system also provides more traditional methods of specifying motion such as commands to move, scale, and rotate pictures, and a path-following command. Although it

relies primarily on 2-dimensional pictures, the system includes a module which can manipulate 3-dimensional pictures.

The versatility and effectivness of the system is attested to by its use. The NRC system was used to prepare several animated sequences for the BBC "Ascent of Man" television series [Bronowski:1973], and a film produced for the National Film Board of Canada by Peter Foldes, a French animator, called "La Faim" [Foldes:1974] received the award of Prix du Jury at the 1974 Cannes Film Festival [Wein:1976].

Recent work involves investigation of "skeleton techniques" [Burtnyk:1976], whereby figures such as an arm that is to bend is associated with a "stick figure" skeleton. The skeleton is manipulated to work out the sequence and then is "fleshed out" in the key positions needed Linear interpolation is used to produce the inbetween frames.

The NRC system is the best of several animation systems to rely on linear interpolation as the primary method of producing animated sequences. Continuing developements promise to enhance an already useful and versatile system.

II.2.4 Real-time Animation

Real-time animation bears a number of similarities to interactive animation. A graphic display and the interactive controlling devices used with interactive systems are also used in a real-time system. The user interacts with the computer directly, gesturing, pushing buttons, moving dials to "show" the computer what is to be done, and observing the effects of these actions as displayed immediately. The primary difference between the two forms lies in the manner of control used.

In an interactive system, the buttons, dials, etc., are used to demonstrate what the computer is to do. The actions required are recorded in some form of agenda and, at a time chosen by the user, these actions are carried out resulting in the display of the film sequence. In a real-time system, there is no agenda. Pictures are created and some of their parameters (such as position) are associated with the interactive control devices. A film sequence is made by manipulating the control device (which causes the picture to change). Action and effect are simultaneous.

SCANIMATE

SCANIMATE [Honey:1968, 1971] is not strictly a computer animation system in the same sense as the other systems that have been described. The other systems store, manipulate and draw the pictures used; SCANIMATE only manipulates them. However, SCANIMATE is an important computer-based animation tool that deserves mention.

SCANIMATE uses an analogue computer to process a television image signal. Pictures are prepared as normal static artwork and placed in front of a television camera. The signal from the camera is fed into the SCANIMATE computer where it is processed by the analogue circuits, and then displayed on a television monitor or recorded on video-tape. A large number of circuits are available to manipulate the image in a variety of ways; these are selected and controlled by a large set of potentiometers, buttons, dials, etc. SCANIMATE is truly real-time: the picture image is continuously produced by the television camera, processed and displayed; the only changes are those effected by altering the control devices and their effects are immediate. In this form SCANIMATE has been widely used in advertising, promotional films, and for imaginative titling effects.

Experimental work has been done with an enhanced version of SCANIMATE called CAESAR [Holman:1975, Honey:1971]. With this version, separate parts of the television signal can be picked out of the original, separately manipulated, and re-combined for display. This technique was used to experiment with cartoon animation, where the arms, legs, head and body of cartoon characters were the separate parts controlled. One of the more interesting techniques attempted involved use of an "anthropometric harness" - an arrangement of sensors worn by a person. The sensors recorded body movements - the bending of an arm, leg movements and the like - and this information was used as part of the SCANIMATE controls. The sensor's information was used to control the equivalent body parts of cartoon characters so that a person could act out the movements wanted of the cartoon characters. The real-time nature of the system meant that the person in the harness could see the effects he was having on the animated character immediately, and adapt his actions to achieve the desired effects.

SCANIMATE is interesting in the totally different approach to computer animation it presents. For a brief time it became popular with the advertising agencies who were its major users, and (North American) television abounded with advertisements made with it. The flurry died down as the "sameness" of the effects it could produce became apparent. This sameness was difficult to overcome because new functions were difficult to add and because a technician was required to operate the machine rather than the user.

GRASS

A more-conventional real-time computer animation system is the Graphics Symbiosis System (GRASS) developed by Tom DeFanti [DeFanti:1973]. GRASS consists of a graphics language and operating system run on a dedicated mini-

computer. The user can sketch pictures interactively (using a stylus and tablet), then write "Macros" (graphics programs) which associate picture parameters with analogue control devices. Executing the Macro causes the picture to be displayed; adjusting the controls causes immediate changes in the pictures using the procedures coded in the Macros. The user makes film sequences by preparing a number of pictures and Macros, sets them running, then manipulates the controls to suit the sequence being produced.

In GRASS pictures may be simple 2-D or 3-D rigid objects, or articulated groupings of simple objects. In articulated pictures, individual components may be controlled separately while the group as a whole may be under a separate control. Early GRASS examples included airplanes flying about the screen (group control of position) with spinning propeller and flapping wings (individual control).

GRASS was originally developed as part of Dr. DeFanti's PhD. work. It was then used primarily for "artistic" and educational films. In its current implementation (at the University of Illinois) it forms part of the "Circle Graphics Habitat" [DeFanti:1976b] and is used to make educational films (or video-tapes). A recent use of a different sort has developed: the immediate, real-time response of the system to the actions of the user, coupled with the display capabilities of the associated video equipment [Sandin:1976], have resulted in several "visual concerts". The GRASS system is used as a real-time performance "instrument" in a concert setting rather than to produce films to be re-played over and over again [DeFanti:1977].

GRASS appears to be a successful attempt to overcome the main limitations of GENESYS. The user-written macros provide the user with access to the computational capabilities of the computer for the algorithmic generation

of sequences. User-written macros may call other user-written macros and so new "commands" may be added to the original set; and recent developments have added a shape-change capability to the system [DeFanti:1977].

II.2.5 Recent Developments

Two recently reported animation systems reflect the current direction in which computer animation systems are developing. The trend is toward the use of colour raster displays with video recording facilities. These systems tend to depend on specific hardware and tend to be costly (at present).

SHAZAM

[Baecker:1975] describes a "conversational extensible system for the animation of shaded images" (p.32) developed at the Xerox Palo Alto Research Centre in 1974. SHAZAM shares a number of common attributes with GENESYS - fixed, user-prepared pictures ("cels") using picture selection as the main tool for shape change. SHAZAM pictures, though, are coloured raster images rather than the dots of GENESYS, giving a much-improved image. SHAZAM includes a superb picture-sketching sub-system.

Three characteristics of SHAZAM are of note: the basic graphic components are "cels" and "movies". A cel is a single static picture while a movie is a sequence of cel selections. Movies may be combined to form more complicated sequences (with the components overlapped in time and space) and recorded on video-tape. Concurrency is another essential feature of SHAZAM. As an example, a movie may be activated and displayed in one corner of the display while its cels are being prepared. The movie will continuously cycle through its cel selection until stopped. The animator preparing a cel for a movie can see it in the context of his movie simultaneously with the sketching. Interaction with

the system is handled largely through use of a gesturing device and menus. Most conventional menus use words to represent commands; SHAZAM uses icons (simple pictures). Finally, SHAZAM is extensible. Users may readily add new features to the system in a particularly easy manner. The ease with which extensions may be made is a direct result of the language used (SMALLTALK [GOLDBERG:1976]), and the clear concise design of the system.

SHAZAM was designed to provide an extensible animation language for use by children and has succeeded admirably [LRG:1976]. Like all well-designed things for children (toys, games, learning material etc.) it is of equal interest to adults. SHAZAM relies on sophisticated hardware (including a video disk) and software (SMALLTALK) and uses both to excellent advantage.

The Cornell University Systemm

[Levoy:1977] describes an animation system under development at Cornell University. The over-all system consists of several separate software sub-systems and two computer systems - one, a large-scale mini-computer with a sophisticated vector display, and the other a smaller mini-computer with colour raster display.

In use, the system is interactive, with the vector display used as the main display and the colour raster display for recording purposes. Pictures may be prepared for either vector or raster displays and routines are available to transform pictures from one form to the other. Sequences may be developed and previewed in "real time" on the vector display using simplified images, then recorded in a stop-frame manner (frame-by-frame) using the raster display. In a manner similar to the NRC system, there is a limited capability to work with 3-D pictures; most pictures are 2-D. As with the NRC system also, the main animation technique is picture interpolation.

There are two significant features of the Cornell system. The ability to transform freely between picture display schemes (vector and raster) and to combine pictures from both schemes seems well thought out. Other systems have translated vector images to raster format for display purposes rather than as an integral aspect of the system.

The second and most significant feature is the "multi-plane" animation technique. With this technique, animated sequences may be described to occur on each of several "levels". The animation on one level is independent of that on other levels but may occur simultaneously with them. Each level is defined as being a specific distance (in the Z direction) from the "camera" (the 2-D display surface). This distance determines the size of the portion of the level that can be displayed (the display window) and scales the motions on each level appropriately; pseudo-perspective effects can be acheived similar to those of the Disney Multi-plane camera of the late 1930's [FINCH:1975] (which inspired the feature). The Cornell system also allows manipulations of the levels themselves well beyond the scope of the Disney device.

The Cornell system is new and unproven. The one article describing it has presented a hodge-podge of techniques and ideas with sometimes tenuous connections between them. It will be of interest to see how the system develops.

II.3 Motion Control Techniques

The aspect of computer animation that distinguishes it from other branches of computer graphics is need for, and methods used to specify and control motion. Some of these methods have been alluded to in the previous discussion.

The following discussion concerns the ways that have been used to specify and control motion in previous

animation systems; it does not discuss the ways motion may be acheived. Anything that can cause an apparent change from one frame to another can produce motion. A "fade" for example is the change in the brightness of a picture and is considered here to be a "motion". The number of ways that motion may be accomplished is limited only by human imagination.

Five basic techniques have been used to specify motion in previous animation systems.

(1) Incremental Change - For each aspect of a picture (or composite image) to be changed, the amount of change required for each frame is specified. For example, to move a picture in the X direction, a MOVE command may be used specifying an increment of change (say 1/2 unit). A controlling loop is then needed to apply this increment for each frame of the sequence. The total amount of motion (distance moved) is a product of the increment specified, and the number of frames produced. The controlling loop may be part of the animation "system", or may be coded by the animator directly.

(2) End-point Interpolation - For each aspect of a picture to be changed, the beginning and final values are specified for a sequence of stated duration. The "system" determines the amount of incremental change necessary for each frame and provides a mechanism for doing the "looping" automatically. The beginning values may be explicitly stated ("MOVE the picture from HERE to THERE"), or may be implied ("MOVE the picture to THERE" (from wherever it currently is)).

(3) Linear Interpolation - This technique may be viewed as a specialized form of end-point interpolation, where each point in a picture is given a specific final position or, as is most common, may be viewed as a separate technique. The algorithm is used to transform a picture's shape from its

beginning shape to the shape of a second picture over a specified time interval. The animator supplies the beginning and ending pictures, and the "system" computes the shape of the intermediate pictures. The fundamental operation of the algorithm is based on each point in the beginning picture having a corresponding point in the final picture. The points of the intermediate pictures are determined as a proportion of the distance from the beginning to the final positions of corresponding points. Linear interpolation is a very versatile technique for specifying motion, but can only be used where the desired motion can be treated as a shape change (a MOVE can be specified as the interpolation of a picture at one location into a picture of the same shape at another location; a FADE cannot unless intensity is treated as part of the picture's shape).

(4) Path Following - The first two techniques may be classed as "symbolic" specification techniques - the motions are specified by symbols (numbers). Linear interpolation may be called a "graphical" technique of motion specification since two pictures are used to specify the motion. Path following is another "graphical" technique. The "path" used is a picture, and some aspect of it - usually its shape - is used to control some other aspect of the picture that is to exhibit the motion. The simplest and most obvious example is the control of a picture's position. The path is treated as a diagram of the sequence of positions that the controlled picture is to take during the specified interval.

(5) Picture Selection - Baecker, in his GENESYS system, used two primary techniques for motion specification - path following (he called these paths "P-curves"), and picture selection. Picture selection is nothing more than the substitution of one picture for another with no consideration for how the second picture was created (in GENESYS each distinct picture had to be hand-drawn).

Baecker "formalized" picture selection through the use of a "selection graph" - a list specifying sequences of pictures to be used. In other systems, picture substitution is much less formal. Animators are free to add or remove pictures from the film frame whenever they wish, but must do so explicitly.

A particular animation system may rely on one of these methods of motion specification, but may provide others. The NRC system for example, relies on linear interpolation for most of its motions, but has provision for some forms of end-point interpolation and path-following. In GRASS, the main technique used is a variation on path-following (the "path" is the sequence of values entered by the user via the control devices), but incremental change is available through the user-written Macros, and a form of linear interpolation has recently been added.

II.4 Uses of Computer Animation

Some of the uses of computer animation have been alluded to above. The following section will discuss several of the ways that computer animation has been used by others, and will discuss those applications for which it is best suited. That computer animation has been found to be a useful tool is apparent from the number of films made using the technique. In 1975, Richard Speer compiled a list of films that had been made in whole or in part with computer animated material [Speer:1975]. The list contains 228 titles and gave 90 adresses as "sources" of computer animated films. Many films were not listed, and undoubtedly many more have been produced since then.

Computer animation can be used for any of the purposes of conventional animation: educational films, promotional films, advertising, entertainment, and art. Computer animation has been used in all these areas. Several series

of educational films have been produced, such as the "Explorations in Space and Time" series [Speer:1975], pg.35. Scanimate was heavily used to make sequences for advertising. The Peter Foldes film "La Faim" was distributed (in Canada at least) as an entertainment short, while several short sequences of computer animation appeared in "Star Wars". A large number of films have been made for "artistic" purposes (Speer lists 51 as specifically artistic); the "Matrix" films of John Whitney and those of Lillian Schwartz are perhaps the best known. Most of the remaining films on Speer's list may be classed as "promotional", where films intended to explain ideas (rather than to teach a topic) are called promotional.

Although computer animation can be used for advertising and entertainment films, it is little used for this purpose. One of the principal reasons for this would seem to be that the image quality is not adequate for these purposes. Most computer graphic imagery tends to be rather stylized - from the simple wire-frame images most vector display systems produce, to the coarse textured images of many raster displays. Where high-resolution, surface shaded, 3-D pictures can be displayed, the costs for single pictures is very high, and the cost of animated film is exorbitant ([Blinn:1976], p. 547, quotes 25-30 minutes of PDP-11/45 processor time for a single picture of modest complexity). The fact that most computer animation systems have been developed in universities and research centers where computer use policies discourage or prohibit commercial activity, is also a factor. These difficulties are not insurmountable, as the continued existence of such companies as the Mathematical Applications Group Inc. (MAGI - a company that produces coloured, shaded, 3-D computer animation and which first published material about its system in [Davis:1968]) proves.

In educational and research institutions the uses are primarily for educational and explanatory films. Numerous individual films, and series of films, have been produced to teach or supplement the teaching of topics, primarily in the areas of mathematics and science [Hopgood:1974], [Schwartz:1968]. Explanatory films come in many forms, from straight-forward descriptions of ideas and processes ([Baecker:1970] describes the GENESYS animation system), to films made by researchers to help them visualize their work and to help describe its progress.

In the research lab, computer animation has several uses. It may be used to display the results of simulation experiments either in representational form (with pictures that look like the objects involved), in iconic form (where symbolic pictures are used to represent objects or ideas), or as statistics (eg: time-varying graphs) [Levine:1975]. It may be used to aid the understanding of complex processes ([Baecker:1973], or as a means of representing things that are too large, small, fast or slow to be otherwise displayed (eg: chemical reactions at the atomic size and time scales, or galaxy formations at astronomical size and time scales). It may be used to "show" things that have no physical form (eg: 4-D hyper-objects [Olshevsky:1970]). In most cases conventional animation techniques could be used to make equivalent films, but the delays involved, the skills and equipment needed, may not allow it. Where precise calculations are involved (such as a film of the operation of a detailed mathematical model), computer animation may be the only possible technique.

Art films are produced almost everywhere that computer animation systems exist. These films may be the result of specific projects ([Mezei:1973], the "Matrix" films of John Whitney), experimentation by people interested in film-making, or may be a by-product of other activities (the "Dynamics of Disk Galaxies" by Frank Hohl can be equally

viewed as the results of a simulation experiment, or as a work of art).

An interesting application of animation techniques is a film analysis system called GALATEA, developed at the University of Chicago [Futrelle:1974]. In this system, computer displayed dynamic images are encoded from live-action film segments in order to analyse the moving images there.

Not all types of animation system are equally well suited to all application areas.

Batch animation is excellent where the movements or shapes of the pictures must be precisely controlled. Explanatory films showing the results of computer simulations, models, or data analyses are best made with batch systems because the computer can do the computations involved then use the results in the animation directly. Educational films sometimes benefit from this approach, particularly where a series of films is to be produced. The modular design of the program for one film often means that the modules may be re-used for other films in the series. In other situations the ability to exactly re-create a sequence (by re-running the original program) may mean that refinements to the film can be better controlled.

Interactive and real-time systems are more "fun" to use and generally require that fewer, less-abstract skills be learned by their users. They are thus more readily available to a wider range of potential users. They also tend to be less precise - the user generally cannot draw on the computational capabilities of the computer in the manner available to the users of a batch system. Repeatability is also more of a problem because, generally, the user must repeat some action (a gesture perhaps), rather than re-use a specific program. For films that must simply "look" right (rather than be precisely correct), the interactive systems

have some decided advantages. The primary one is the ability to create and refine sequences with very minor delays.

Chapter III

The Design of an Appropriate System

It is tempting to design a computer animation system that meets (or attempts to meet) lofty ideals, and then to attempt to implement part of the designed system. The process is not quite so direct and simple: design ideals are quickly transformed as trade-offs are encountered between the ideal and actual practice. Alternatively, one can retain one's ideals, create a design, then attempt to buy or build the hardware and software to realize the design - an option that is not available to most people.

This chapter presents the design of a computer animation system "appropriate" to the local environment. It attempts to do so independently of the actual implementation details (which are presented in Chapter IV); however, some of these implementation details do creep in to explain and justify some of the design decisions, or to explain why some design features receive only cursory attention. The design is that of YACAS as it has been implemented. A number of additional features that have not been implemented are alluded to in this, and the next two chapters; this is occasionally necessary to explain design decisions. Chapter VI discusses these features as part of the grander design of YACAS.

III.1 Design Objectives

The design objectives for the system are fairly simple and straight-forward:

(1) The system must fit into the local environment.

(2) It must be available to a wide range of users. Both users with computing experience and those with none should be able to use the system.

(3) The animation system should interface to other research activities (particularly those which make use of computing services).
(4) It should be efficient in both memory and processor utilization.
(5) It should provide "tools" to aid the documentation and debugging of the "films" produced.
(6) It should simplify the process of describing complex motions, and of describing motions of complex pictures.
(7) It should be extensible.

III.2 Fundamental Considerations

In arriving at a final design for an animation system, a number of major items need to be looked at. These provide a framework for the rest of the design.

III.2.1 The Environment - Hardware and Software

A computer animation system (or any system) may be designed with little consideration of implementation restraints, and then adapted to suit the hardware and software available. The design of an appropriate system needs to take such matters into consideration from the start.

The hardware environment existing at this installation (the University of Canterbury's Computer Center) consists of the following: a Burroughs B6718 processor with 160K (48 bit) words of main memory, disks, magnetic tape, paper tape reader and punch, a card reader and punch, line printer, 10 inch drum plotter, several terminals including VDU and printing types, a Data Communications processor, and 3 RJE stations. This machine operates under the Burroughs MCP operating system with ALGOL and FORTRAN as the main languages, and compilers and interpreters for PL/I, COBOL, SNOBOL and other languages. An "online" system (CANDE) is

available on a limited basis for conversational programs. The graphics support for the plotter is rudimentary.

An interactive graphics facility is available as a separate machine. It is a Digital Equipement Corp. PDP-11/40 with 28K words of main memory, a VT11 display processor and 17" vector graphics display, 2 RK05 cartridge disks (about 5.4 MB storage), and an LA36 Decwriter - a standard GT44 system. This system uses the RT-11 operating system (single user), with FORTRAN and MACRO (the PDP-11 assembler) as the main langauges. BASIC is available. The graphics support is that supplied by the manufacturer.

There is a communications link between the PDP-11 and the B6700 supported by software of limited flexibility. Files may be transmitted from the B6700 to the PDP-11 provided they are in a specific format (card images). The content of the files is immaterial provided the proper operating procedures are followed. Only files destined for printing may be transmitted from the PDP-11 to the B6700, and these only during the CANDE sessions.

The B6700 is operated as a batch machine available to users for 9 to 12 hours daily, 5 days per week. The main magnetic bulk file storage medium is magnetic tape; user access to on-line disk file storage is available in limited amounts. The PDP-11 is operated on a 1/2 to 1 hour elapsed-time, pre-booked basis, with limited access outside normal hours (10 AM to 5 PM).

III.2.2 Batch vs Interaction

An interactive animation system was ruled out, partly for hardware reasons and partly because a batch system will meet several of the objectives of III.1 better than an interactive system would.

Although the GT44 is intended to be an interactive graphics facility, it is only marginally able to be so used. Good, effective interaction requires a variety of methods for the user and computer to use for communicating with each other; the GT44 has only a keyboard, display, and lightpen. This set of devices forms the bare minimum for interactive graphics. The lightpen, potentially the most useful device for the user, is badly designed (it has no enabling switch) and would frustrate effective interaction.

A batch system is better suited to the needs of the research worker than an interactive system would be. A batch system can usually make better use of the computational capabilities of the computer to control the film produced. The result can be a film that is more correct in detail than is possible with an interactive system (where the appearance of a film is more easily controlled than the details). Where large volumes of data are to be "animated" in some way, a program is better able to process the data with fewer mistakes than a person using an interactive system.

III.2.3 Language

Having a batch system implies that the user will need to write a program in order to make a film. A question arises as to the nature of the language used. It could be a language specifically designed for the purpose, it could be a set of new statement types added to an existing language, or it could be a set of subroutines callable from one or more standard languages. The subroutine approach - with the subroutines written in Burroughs Extended ALGOL - is the one that has been adopted.

This approach was chosen for two reasons: (a) most users would not need to learn a new specialized language, and (b) implementing a set of subroutines is easier than writing a language translator or modifying an existing one.

ALGOL and FORTRAN are the two principle languages available for use as the implementation language. ALGOL was chosen because (a) it has better program structuring facilities than FORTRAN, (b) it is widely used, and (c) ALGOL subroutines may be called from a FORTRAN program (thus allowing a FORTRAN programmer to use them).

Some consequences of this decision are discussed in Chapter VI.

III.2.4 Graphics Aspects

In the current implementation, the system uses 2-D vector graphics producing black and white "wire-frame" images. The over-all design includes facilities for surface shading, 2 1/2-D hidden line removal, and colour added through optical techniques. These aspects are not implemented yet; "handles" are provided in the data structures to allow them to be added at a later date; they will be discussed further in Chapter VI.

Two considerations influenced these decisions. (a) The hardware available is exclusively vector oriented; a system based on raster graphics would be inappropriate. (b) Ease of implementation was the primary motive behind the selection of 2-D graphics; 2-D is also suitable for a wide range of film subjects (conventional cel animation is entirely 2-D), and is more easily understood than 3-D by inexperienced animators, especially where the ability to interact with one's work is restricted.

III.3 The Design

This section presents the design of the system. Where possible, it attempts to do so without reference to the details of how the features are implemented; these details are discussed in Chapter IV. Where implementation details must be mentioned, the discussion is brief, with more detail

presented in Chapter IV. The discussion begins with an overview of the features of the system, followed by a more detailed discussion of its major aspects.

III.3.1 Overview

Certain fundamental aspects of the design have already been mentioned: the system is a batch one, implemented as a set of ALGOL subroutines to be called from a user-written program. Pictures are 2-D, black and white, "wire frame" images.

YACAS consists of two components. (a) A set of subroutines to provide a user-written program run on the B6700 with the ability to create a "film" in the form of a file. (b) A program, called PLABAK, which is run on the GT44 to process this file and produce the sequences of images on the display screen as moving pictures. These pictures may be recorded from this display to produce the animated film. The two components reflect a functional separation: the subroutines are used to produce the "film" in an intermediate form which PLABAK is able to display.

The set of subroutines includes the following features.
(1) The ability to describe, store and display pictures.
(2) The ability to associate pictures in a hierarchical group called an "articulated picture", and to manipulate the group as a single picture.
(3) A set of commands to manipulate the pictures statically and dynamically.
(4) The ability to specify motion dynamics.
(5) The ability to "display" pictures on either a film file, paper plotter, or both.
(6) The ability to extend the system with user-written motion commands and dynamics functions.

PLABAK provides the user with the following functions.
(1) The ability to control the rate at which a film is

displayed.
(2) The ability to view individual frames of a film and to step through a film frame-by-frame.
(3) The ability to specify and display sub-sequences of a film.
(4) The ability to override the display window specified in the film, substituting a user-specified one.

III.3.2 Pictures, Cels, and Articulation

The basic elements of the system are the pictures the system can manipulate. YACAS makes a distinction between the shape or appearance of a picture, and other attributes such as position, size, intensity, etc. This distinction is reflected in the existence of two types of picture elements: "cels" which define the shape of an image, and "pictures" which define the other attributes.

The word "picture" will be used here with two main meanings: (a) occasionally it will refer to the abstract attributes that are dissociated from the shape (where confusion might arise, the expression "picture-header" will be used), (b) the combination of shape and attributes that may be displayed will be the usual meaning of the word (where necessary the expression "displayed picture" will be used to make the distinction).

An individual picture (picture-header) may be associated with only one cel; a single cel though, may be associated with more than one picture. Thus there is a many-to-one mapping from pictures to cels; each picture is an instance of a single cel. A cel cannot exist without an associated picture. Pictures may exist by themselves (and are termed "simple"), or may be associated with other pictures in a hierarchical arrangement (termed "articulated"). A more-complete description of these distinctions may be found in the User Guide, section 2.1.

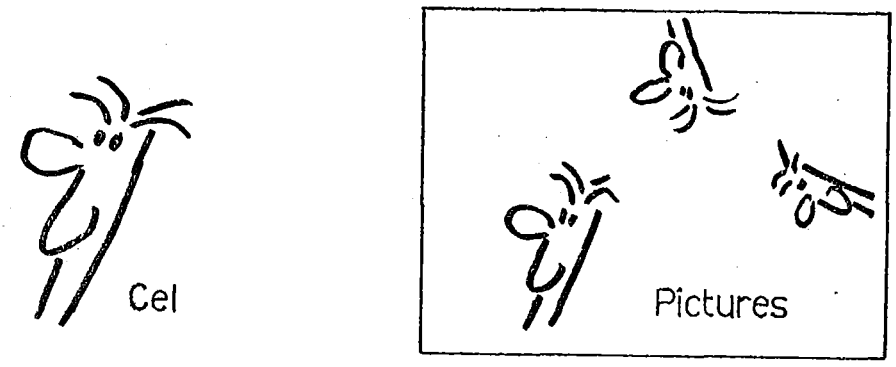


Fig. III-1
A Cel with several pictures (instances).

The attributes "contained" in the picture-header include the following: position, scaling factor, orientation, and intensity code. The vectors making up a cel have some inherent position and size based on the coordinates used. The position inherent in this description is removed when the cel is first described and stored in the picture header of the associated picture. The cel description is "normalized" so that the coordinates are given relative to a local origin called the "pivot point" (at (0,0)). The inherent size of the cel is not altered. Instances of cels can have different sizes by scaling the cel description according to the "scaling factors" stored in the picture header. There are two scaling factors to allow scaling in both the X and Y directions. When the cel is first described, these factors are set to 1. A cel description has no inherent orientation, and the picture header "orientation" factor is initially set to 0 degrees.

Other picture-header information includes a link to the associated cel (there is no reverse link) and links between associated pictures in an articulated picture group.

A cel defines the shape of a picture. Each cel is composed of three sections of vectors plus a small amount of "header" information. The three sections are the detail,

outline, and attachment point sections. The detail section contains the main descriptive information about the cel. The outline section contains vector list descriptions of one or more closed polygons which constitute the cel's outline; polygons contained within other polygons represent "holes" in the outer polygon. The attachment point section contains a set of non-visible points, considered and manipulated as part of the cel, to which son-pictures in an articulated picture may be attached. An individual cel need not have information in each section, but must have information in at least one of its sections.

An articulated picture is a group of simple pictures arranged in a hierarchy. One picture is the "root" and may have an arbitrary number of "sons"; each of the sons may itself have an arbitrary number of sons. The group may be manipulated as a whole (treated as a single picture), or individual pictures may be manipulated separately. Pictures in an articulated group are "attached" to the picture above them in the hierarchy. Attachment is accomplished by binding the position of the "son" picture's pivot point to one of its "father's" attachment points.

Fig. III-2a shows a simple articulated picture having one root and four "sons"; fig. III-2b shows the hierarchy. A more complicated picture could be produced by subdividing the arm into an arm and hand, and the leg into a leg and foot (fig. III-2c).

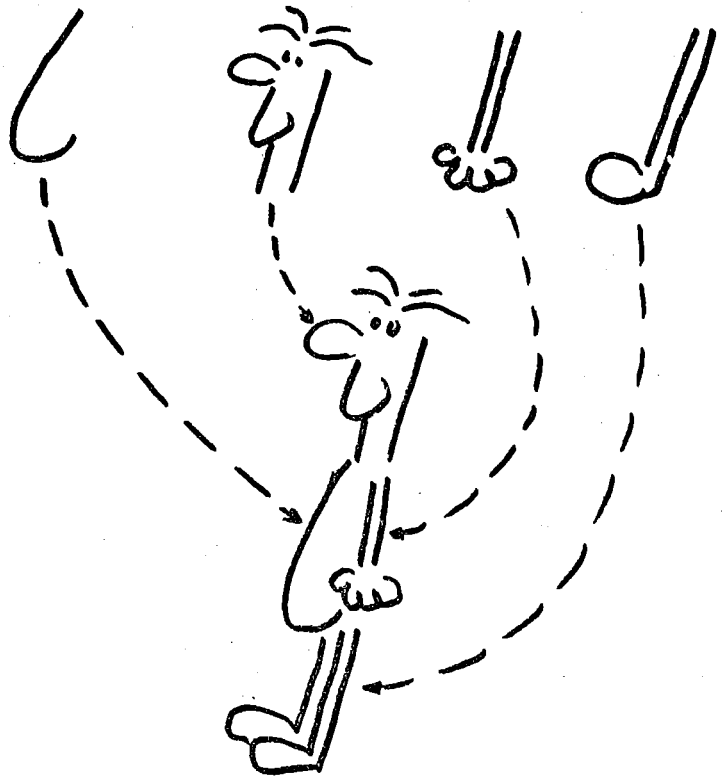


Fig. III-2a
An articulated picture; its components and assembled.

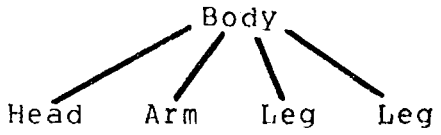


Fig III-2b
The picture structure of III-2a.

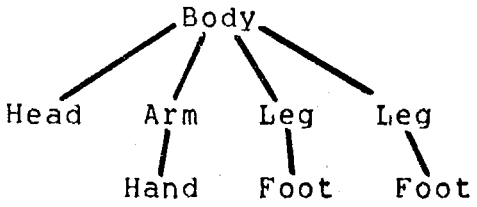


Fig III-2c
A more complicated structure.

This scheme for handling picture storage was chosen for the following reasons.

(1) In situations where one picture shape is repeated several times in a frame (eg. a chemical model with a single shape for an atom), the vector list defining the shape needs to be stored only once. Other animation systems commonly treat each picture as a separate entity,

duplicating the shape vectors for each picture.
(2) Linear interpolation is more effectively integrated with the other ways of producing motion where shape is separated from the other picture attributes. Where no separation exists interpolation not only changes the shape of the picture, but may also change its position and size. Separating shape information from the other attributes means that interpolation will affect only the shape, allowing separate control of the other attributes.
(3) The non-distorting transformations (MOVE, SCALE, ROTATE, FADE) are more easily applied to a picture since they affect only the picture header values rather than each point in the picture description.
(4) The ability to create articulated pictures makes the description of complicated motions (a walking character, for example) easier to specify. To have the character walk, the swinging motions of its arms and legs need only be specified as though the character was walking on the spot. The rest of the character is moved separately, independent of the swinging motions.

The separation of picture shape and attributes may be more costly in terms of picture storage space than would be the case where there is no separation. With no separation, the position, size, and orientation values are intrinsicly part of the picture shape information and do not have to be stored separately. On the other hand, where a cel is used by more than one picture, memory is conserved by not duplicating the shape.

In a batch processing environment, the ease and immediacy of direct free-hand sketching of pictures is not available. More attention has to be paid to the preparation and planning of the film and its graphic elements. Part of this planning includes the preparation of picture vector lists independent of the animation program. These need to be made available to the program and two mechanisms are

provided: (a) picture vector lists may be input via card files (see the discussion of READCEL in the User Guide), or (b) they may be included as part of the user's program through calls to the commands VECT, ARC, and POLYGN.

III.3.3 Motion - Concurrency and Dynamics

Motion occurs when some aspect of a picture changes from frame to frame. It is not the purpose of this thesis to attempt a discussion of all the ways that motion may be achieved - the range of possibilities is limited only by human ingenuity. Nor is it the purpose of this thesis to try to specify some minimum set of methods; in a batch animation system the set of motion commands should be as large as possible to give the user a wide choice.

YACAS in its present form has not implemented as wide a set of motion commands as would be desirable due to constraints on the time available to program them. In partial recognition of this deficiency, a method is available for the knowledgable user to add his own to the set provided (discussed in III.3.6).

Three techniques are provided by which motions may be achieved: picture transformations, linear interpolation, and window transformations. Picture transformations are represented by the commands MOVETO, SCALEBY, ROTATETO, FADE and DISSOLVE. These commands change picture header values from the initial values in the picture header, to the values indicated as the command parameters. The changes are effected through the technique of "end-point interpolation" discussed in Chapter II. The shape of a picture may be changed via the linear interpolation algorithm, available as the command INTERPOLATE. This command affects only the shape of a picture (ie: it operates only on the picture's cel). The third motion technique, window transformation, creates apparent motion by changing the display window; other animation systems occasionally call these motions

"camera" motions. Window transformations are supplied through the commands PAN and ZOOM.

Motions take place over time. In YACAS, the period of time for a motion is called an "interval" and may be of arbitrary length. To specify a motion for a picture, the motion command must be associated with an interval.

The description of an interval is begun with the issuing of an INTERVAL command. Parameters of the command give the starting time and duration of the interval. This command is followed by an arbitrary number of motion commands describing what is to be done in the interval. The end of the interval description is indicated by the issuing of either another INTERVAL command to start the description of a new one, the ROLLIT command to begin processing of the agenda (see below), or the SETSTATIC command which switches YACAS back to "static" mode (see below).

Concurrency

In an animated film, it is rarely the case that only one motion is happening at any given instant. Usually several things are happening concurrently. In YACAS concurrent motions may be freely specified by specifying arbitrary numbers of motions for individual intervals, and by specifying arbitrary concurrent intervals. Each interval is independent of the other intervals and may start and stop at any time. Figure III-3 shows an example of the specification of several concurrent intervals with concurrent motions, and a diagram showing how they overlap in time.

It is important to understand that the INTERVAL and motion commands have no immediate effect on the pictures involved. Rather, the information about the motions and intervals that have been specified is stored in an "agenda" and is used later to direct the creation of the film file. The processing of the agenda in initiated by the user

through the ROLLIT command. ROLLIT processes the agenda for a stated time interval (given as the command's parameters). Any motions that are completed during the processed interval are removed from the agenda; any that are not completed remain, to be carried out by a subsequent call on ROLLIT.

```
INTERVAL(0,5);
 MOVETO(PIC,X1,Y1);
 SCALEBY(PIC,SX,SY);
INTERVAL(2,6);
 INTERPOLATE(PIC,PICA);
INTERVAL(0,10);
 PAN(DX,DY);
```

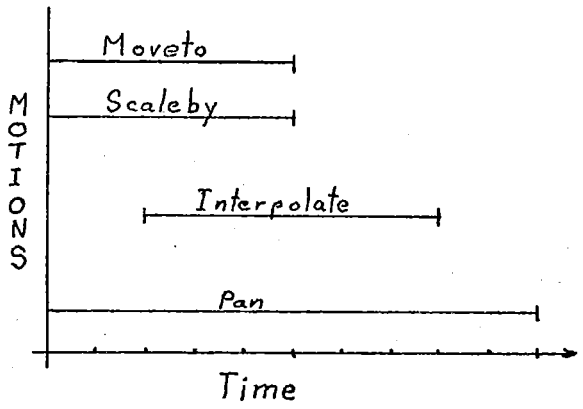


Fig. III-3
a. The specification of several concurrent intervals and motions.
b. A diagram showing how the motions and intervals overlap in time.

Thus motion is specified through the use of the supplied motion commands or through use of user-written commands. These commands are carried out over specified intervals. Concurrency is achieved by allowing an arbitrary number of motions to be associated with a single interval, and an arbitrary number of intervals to be specified independent of each other. The calls to INTERVAL and the motion commands construct an agenda of things to be done. The agenda is processed by the ROLLIT command to produce a film file.

Dynamics

The rate at which a motion is carried out, and particularly the way the rate varies, is called the motion's "dynamics". The simplest dynamics are "linear" dynamics in

which the rate of motion is constant. YACAS supplies two other forms of dynamics - "cushioned" and "sinusoid". These are discussed in the User Guide, section 2.4.

As with the motion commands, the set of dynamics supplied forms a rather arbitrary and small set. A larger set of dynamics commands would be desirable. The restrictions inherent in the small set are partially alleviated by the ability of the user to supply his own (see the User Guide section 5.4).

Each form of dynamics is provided by YACAS in the form of a function, whose value is used by ROLLIT and the motion commands to determine the amount of change needed in each frame of an interval. All of the motion commands implemented condition their actions according to these functions.

The user associates a particular dynamics function with a particular motion command by execution sequence; the dynamics function must be called immediately after the motion command it affects. For example, the sequence of commands

```
INTERVAL(0,12);
MOVETO(PIC,12,8); CUSHIONED;
ROTATETO(PIC,-135.); SINUSOID(1);
```

produces an interval lasting 12 time units, with two motions to be carried out during the interval. One motion is to move the picture PIC from its current position to (12,8) with cushioned dynamics. The other motion is to have the picture (PIC again) rotate to an orientation of -135 degrees with one cycle of sinusiod dynamics.

As with the motion commands, the dynamics commands merely add information to the agenda to show which form of dynamics have been chosen.

III.3.4 Film Making and Evaluation

For a description of how to make a film using this system see Chapter V and the User Guide. The following provides a very brief overview to tie the pieces described above together.

The steps to follow might consist of these:
(1) The film should first be designed, usually via a storyboard, and the pictures to be used identified, designed, prepared and perhaps punched as a card file.
(2) A program is written which describes the film. This program may create pictures through explicit command execution (VECT etc), or by reading prepared descriptions from cards using READCEL. It then specifies the motions and dynamics required for the film. The result of executing this program is a film file.
(3) The film file is copied to the GT44 and previewed interactively on its display using PLABAK.
(4) If the film has "bugs" in it, the program can be modified, re-executed and the new result previewed. This process can continue until the film is correct. When it is, PLABAK may be used in "recording" mode, along with a camera, to actually produce the film.
(5) If the film is long (say, more than 1 minute), it may be split into several "scenes" with a separate program written for each. These separate scenes may then be edited together to produce the final film.

To assist the user in debugging his film, both from the visual side and from the programming side, the system checks command parameters for "legality" (eg. checks to see that pictures it is told to use do exist) and prints warning messages when there are errors detected. The system attempts to continue despite errors of this sort. Another facility of the system is a "cartooning" mode of paper plot output. Selected frames from the film are drawn on the paper plotter giving hard-copy "snapshots" of the progress

of the film. These plotted frames may be scaled such that 1, 2, 3 or 4 of them fit across the width of the plotter paper saving paper and time, and making it easier to follow the progression of the film. A sample of paper plotted output appears in Appendix B.

The Film Previewer - PLABAK

To inspect the film produced by an animation program, the film-file must be transferred to the GT44 where it may be displayed by the film previewing program (PLABAK). This program has been implemented in only a very rudimentary form, but does include the following features:

(1) The ability to preview film segments at close to "real time" speed.

(2) The ability to change the playback speed by altering the single-frame display time (changing the number of frames displayed per second from the standard 24), or by determing a selection criteria for existing frames (eg, displaying each frame 2, 3, or more times; or displaying every 2nd, 3rd, etc, frame).

(3) The ability to step through a sequence on a frame-by-frame basis.

(4) The ability to over-ride "camera" controls by specifying a fixed window to be used as an alternative.

(5) The ability to interrupt the playback at any time to change display parameters.

The previewing system is also used for the recording operation. Recording is accomplished by setting a movie camera in front of the display and photographing each frame of the film while previewing it at the frame-by-frame speed. Synchronization must be performed manually at present. All of the functions available during preview are available during recording.

III.3.5 Film Files

A user's animation program is, in effect, a description of a film. This is translated by the system into the form of a film, ultimately, but passes through at least one intermediate stage - the film file. Commonly a film file consists of complete vector descriptions of each frame of the film; the film files of YACAS are rather different. They consist of picture header descriptions, cel descriptions, and lists of the numbers of pictures to be displayed in each frame. The picture header and cel descriptions are included in the file only when there has been some change in one or the other. The vector-list form of film file used by other systems is usually based on the assumption that the plotting device has no, or very little, memory or intelligence apart from that strictly necessary to drive the display or plotter. The assumption of YACAS is that the film file is going to be used by a film previewing system implemented on a mini computer that has at least moderate memory resources and processing power.

The main reason for the choise of this approach has to do with the size of the film file; it is claimed that YACAS film files are more compact than the more common vector list form. Since picture header and cel descriptions are included only when there are changes, a great deal of redundant information is removed. When a picture is moved, for example, its shape does not change. By copying the cel description to the film file once, at the beginning of the sequence, the redundancy of including the shape information when describing each new frame is removed. YACAS will only copy the picture header information (which is changing its position values) in this situation. Extra information may be included in the film file that would not be under the vector-list system; codes indicating which picture headers are included, which cels etc (its own low-level commands).as well as vector information that may lie outside the viewing

window.

A second reason for this approach is the added flexibility it provides. The processing power of the GT44 is able to be used to perform functions that cannot be done with the more-common form of film file. YACAS has implemented only one function that specifically uses this added flexibility (the ability to redefine the display window). Chapter VI discusses enhancements to PLABAK which rely on this flexibility.

A third reason is perhaps more telling. The GT44's disks provide relatively little room for large film files. Most animation systems archive film files onto magnetic tape with the result that size is of less importance than here. YACAS film files could be archived on tape using the B6700 system, but the clumsiness of the link between it and the GT44 would make such a scheme awkward (see appendix 2 of the User Guide).

III.3.6 System Extensions

Often, no matter how rich the set of available commands may be, it is desirable to add new ones. To facilitate this a mechanism is provided to allow user-defined motion commands and dynamics functions.

In fact three types of commands may be added to the system-supplied set by the user: "static" commands include those which carry out their function at the time they are invoked. The picture definition commands (OPENP, VECT, ARC, etc) are all static commands. This type may be quite easily added by any user because they affect the data structure directly and immediately.

"Motion" commands are those that are repeatedly invoked by the system so that their effect appears to extend over a period of time (the MOVETO, INTERPOLATE and FADE commands are examples). Adding new commands of this type is more

difficult since there must now be two views of the command: the user's view where the command, when used, simply puts on entry in the agenda and does nothing directly to any pictures; and the "system" view which repeatedly calls the active portion of the command to carry out an increment of the motion.

Dynamics functions may be added to the three basic ones supplied in a manner similar to the addition of motion commands.

These three forms of system extensions are discussed in more detail in section 5 of the User Guide.

Chapter IV

Implementing the Design

This chapter will discuss some of the details involved in implementing YACAS. Not all aspects, nor some of the less important details of the design are discussed; for "nitty-gritty" detail the source code must be referred to. What will be discussed will be general algorithms and data structures used to get the system to work.

IV.1 Data Structures

Apart from a number of simple variables used as counters, pointers etc, there are three main data structures that are central to the implementation. These are the picture storage structure (picture and cel tables), the motion scheduling and queuing structure (the agenda), and the film files. The implementation of these structures is discussed below.

IV.1.1 Picture and Cel Tables

YACAS provides its own "memory" for the storage of picture information. This is implemented in the form of two ALGOL arrays. The arrays are called the "Picture Header Table", and "Cel Table". As the names suggest, the first stores picture header information while the second stores cel information.

Picture Header Table

This is a two-dimensioned array. Each row contains picture header information for one picture; the number of rows limits the maximum number of pictures that may exist at

any one time. Specific pictures are identified by a "picture number"; this is simply the row index of the picture in the table. This number is returned to the user when a picture definition is begun via the OPENP command.

The major items of information contained in a picture's header are:
(1) picture position (X and Y co-ordinates of the picture's pivot point),
(2) X and Y scaling factors,
(3) orientation (angle in degrees),
(4) intensity code,
(5) the picture numbers of the picture's father, son, and brother pictures,
(6) the number of an attachment point in this picture's father's cel (if part of an articulated picture),
(7) pointer to the picture's cel,
(8) displayed picture list pointer (see IV.2.1).

Each row of the array is 8 words wide. Values for the position co-ordinates, scale factors and orientation angle use full words (stored as floating point numbers), while all other values use partial words to reduce memory requirements. Fig. IV-1 is a diagram of a picture header table entry.

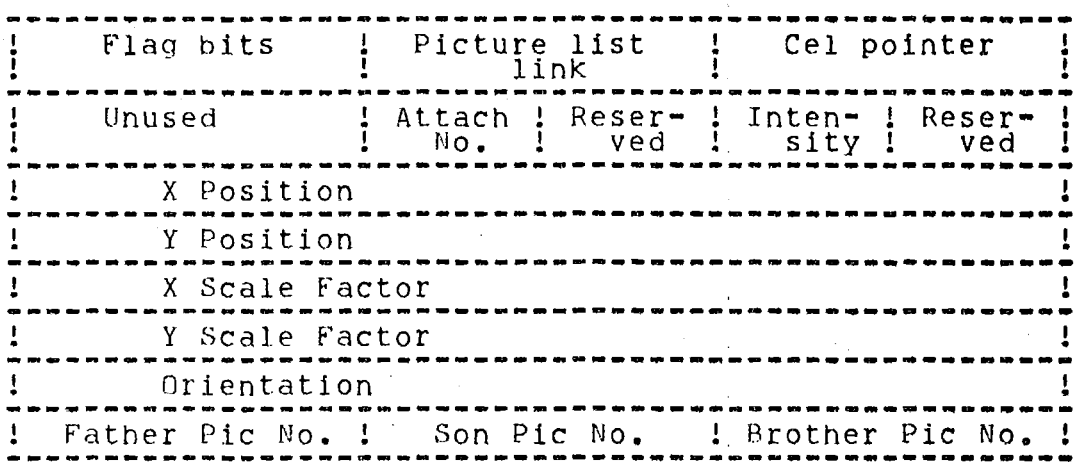


Fig. IV-1
Picture-Header Table Entry

For simple pictures, the space for the father, son and brother picture numbers is unused. When the picture is part of an articulated group these numbers form the links of the structure. Fig. IV-2 is a diagram showing how these entries are used to create a structure consisting of a root picture with three sons and one grandson.

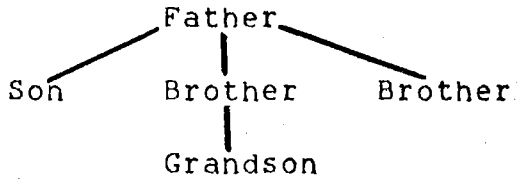


Fig. IV-2
Links of an articulated picture.

Cel Table

This is a single-dimensioned array which contains variable-length cel descriptions. Each cel description consists of a fixed-length cel header followed by three variable-length data sections - one each for detail, outline and attachment point sections. Cels are identified by their "cel number" - the index of the cel's header in the cel table. The purpose of the outline section is to support surface shading and hidden line removal features which have not yet been implemented (see Chapter VI). The present version of YACAS treats outline vectors as though they were part of the detail section. The outline section is not necessary to the current implementation, but is included to make the implementation of the additional features easier.

The cel header contains
(1) a counter of the number of pictures associated with the cel,
(2) total length (number of words in the whole cel entry), and
(3) the offset to the start of attachment point section (to make attachment point lookups easier).

The detail and outline sections contain (a) a header indicating total length and the number of curves in the section, and (b) for each curve, the number of "points" in the curve followed by that number of X,Y pairs.

The attachment point section is only slightly different, containing (a) a header indicating total length and the number of points in the section, and (b) the set of points (not grouped as curves).

This organization avoids the need to specifically store connectivity information with each vector as is commonly done with vector graphics systems. Each curve begins with an invisible vector to the first of its points, followed by visible vectors connecting its other points in the order given. The attachment point section contains only points with no implied vectors (visible or invisible) connecting them.

A diagram of the layout of an entry in the cel table is shown in Fig. IV-3.

How this data structure compares with what others have done is difficult to assess since such details are rarely published. [Williams:1971] describes a variety of data structures used in computer graphics systems, but these are complex structures suited to the design applications they support and are more complex than YACAS requires.

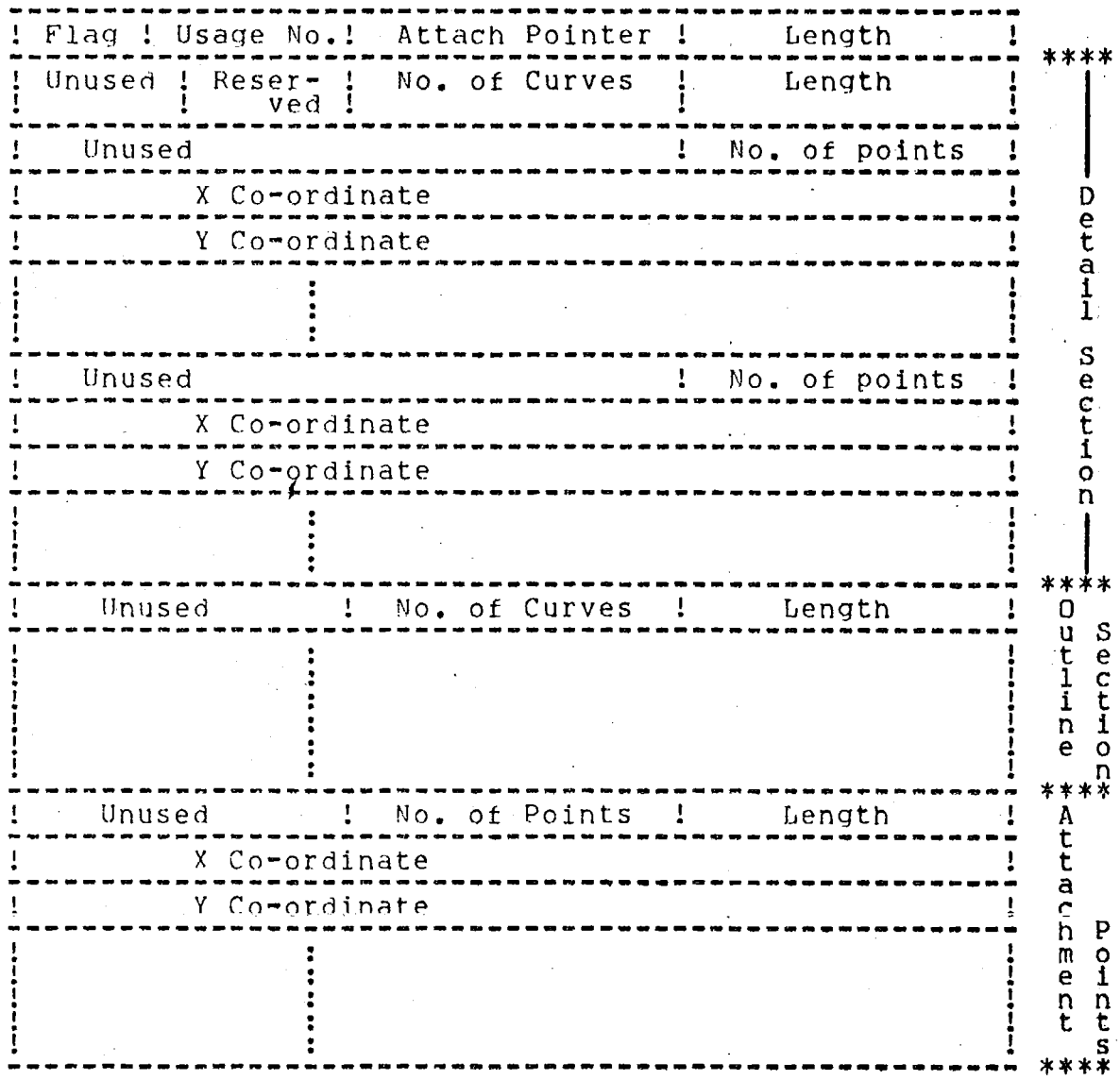


Fig. IV-3
Cel table entry

Memory Management

Under the current implementation of YACAS, the picture and cel tables are fixed-size arrays.

For the picture header table, memory management is trivial: an entry is allocated by doing a linear search through the table until an unused row is found. To release an entry, the row is cleared (each element set to 0.).

Management of the cel table is more complicated since the space is used as variable-length blocks. Free space is maintained as a linked list of variable-sized blocks. When a picture is opened, the largest block is allocated to hold the cel description. When the cel description is complete, unused space in the block is returned to the free list. When a cel is deleted, its space is returned as a block to the free list. When blocks are added to the free list they are added in ascending index sequence, and are combined with any other contiguous free blocks. No "garbage collection" routine has yet been implemented.

IV.1.2 The Agenda

The agenda has two functions: (a) to maintain a record of the actions to be performed, when and for how long, what pictures are involved, what dynamics etc., and (b) to contain a record of the progress of each action as it is carried out. These functions are discussed in more detail in IV.2.2.

An array (two dimensions - one row of 8 words per entry) is used for this purpose. Each agenda entry contains
(1) start and stop "times" (frame numbers) for the motion,
(2) a numeric code for the motion to be performed,
(3) the number of the picture to be acted on,
(4) space for four (ALGOL REAL) parameters for the motion command,
(5) a code for the dynamics function to be used and one parameter for it,
(6) the previous dynamics function value (see IV.2.2).

A diagram of the layout of an agenda entry is shown in Fig. IV-4.

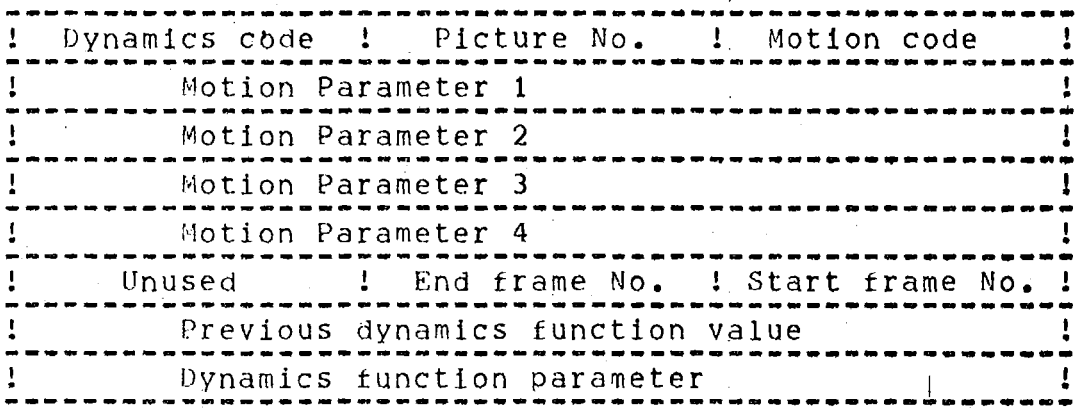


Fig. IV-4
Agenda entry layout

The contents listed are sufficient for the commands implemented so far. One can envision the possibility of needing more storage for parameters for new commands; the array row length can be extended by changing one DEFINE statement without requiring any further changes to any of the other subroutines.

IV.1.3 Film Files

Film files are produced by the user's batch program and are displayed by the PLABAK program on the GT44. Thus, they are used to transfer the film information between the B6700 and the PDP-11, and to store this information for later display.

The information in this file can be thought of as a sequence of "commands" to the playback system. The commands, with their associated data, are variable-length sequences of (16-bit) words. The following commands have been implemented:

0 - No-op

1 - Unused

2 - Advance - marks the end of a frame. Takes a single parameter which is used to indicate the number of times

the frame is to be copied. For example, a parameter value of 5 indicates that 5 identical frames (the just-completed one) are to be displayed (see User Guide section 3.1 - the FILMSPEED command).

3 - Define Cel - marks the beginning of a cel description. parameters: total number of words in the description, the cel number, the sets of detail, outline, and attachment points.

4 - Delete cel - the parameter is the number of a cel to be deleted.

5 - Define Picture - marks the beginning of a picture description. The fixed-size description includes the picture number, associated cel number, intensity, position, scaling factors, orientation (mod 360 degrees), father, son and brother picture numbers, attachment point number, and a word of flag bits.

6 - Delete Picture - the parameter is the number of a picture to be deleted.

7 - Draw Pictures - marks the beginning of a list of picture numbers to be drawn in the current frame. parameters: the number of picture numbers in the list, followed by the list.

8 - End of Film - marks the end of the film.

9 - Define Window - marks the start of a new window description. parameters: the co-ordinates of the lower-left (XLL,YLL) and upper-right (XUR,YUR) corners of the new window.

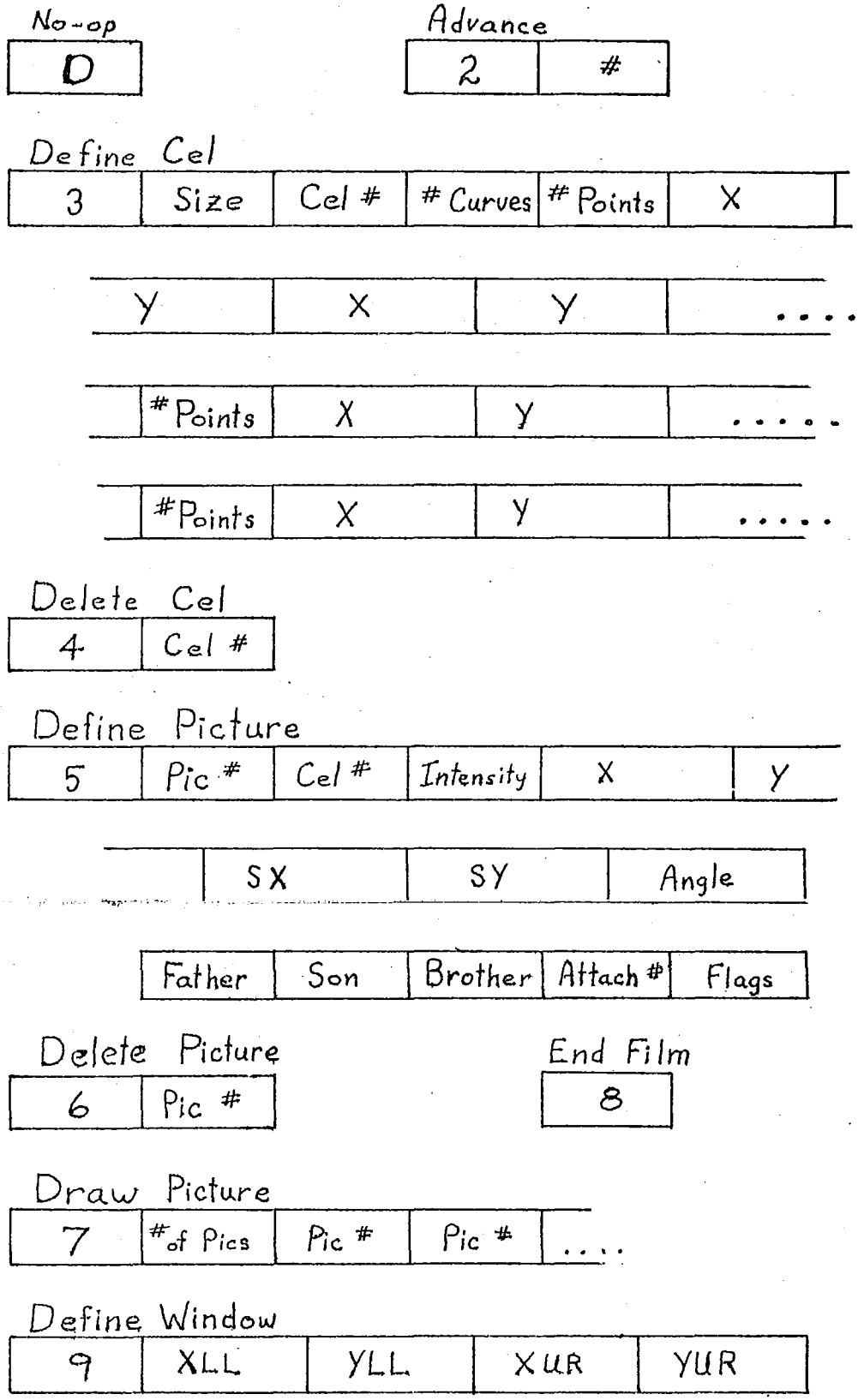


FIG. IV-5
FILM FILE "COMMAND" LAYOUTS

The batch YACAS system produces a film file containing PDP-11 formatted Integers and Real numbers. There can be some loss of precision when converting Burroughs REAL numbers (48 bits) to PDP-11 REAL format (32 bits). Numbers that are too large are treated as the PDP-11's largest value; numbers that are too small are treated as 0. This loss of precision is not thought to be important since the PDP-11 floating point number range (about +/- 10**(+/-38)) is quite large. Integers pose no problems since only Burroughs partial word values (none over 16 bits long) are converted to PDP-11 Integer format (16 bits).

IV.2 Some Basic Methods

This section will discuss some of the methods used to implement the features of the YACAS system. This discussion will center on the implementation of features of the batch phase of the system. Descriptions of the commands mentioned may be found in the User Guide, section 3.3.

IV.2.1 Picture Display

The batch phase of YACAS has three principal functions: to store picture descriptions, to manipulate them, and to display them as directed by the user. Pictures may be displayed on paper via the pen plotter, or on the GT44 via film files (or on both media). The user has two distinct ways to specify which pictures are to be displayed: explicitly with the DRAW command, or implicitly when describing sequences.

Explicit picture display involves the static manipulation of pictures and the explicit indication of which pictures are to appear in the frame using the DRAW command. When all the pictures to be displayed have been mentioned in a DRAW command, the ADVANCE command is used to indicate that the frame description is complete. The DRAWF

command is a short-hand way of indicating that the picture indicated is the last picture of a set to be displayed; the ...F indicates that a frame ADVANCE is to be automatically executed. An ADVANCE issued with no preceding DRAW's produces a blank frame.

Implicit picture display is performed by YACAS as part of the sequence description mechanism. A sequence consists of an INTERVAL command followed by one or more motion commands, followed by either a ROLLIT or another INTERVAL command (see section IV.2.2). The pictures directly affected by the motion commands in the sequence will be automatically displayed by YACAS in each of the frames of the sequence. The DRAW command may be used in a sequence description to force the display of a picture that does not appear in any motion command.

Independent of the mechanism used to select the pictures to be displayed is a choice of output device to be used. YACAS can provide picture display via either film file, the paper plotter, or both. This choice is specified via the ENVIRONMENT command. Once an output device has been selected it may be temporarily made inoperative via the INHIBIT command, and made available again via the RESTORE command.

Postponed Display

Each of the picture transformation commands affects some separate aspect of the picture description. For any particular frame more than one transformation may be applied to a specific picture. To ensure that all transformations on a given picture are complete before the picture is displayed, YACAS postpones the "actual" display of the pictures in a frame until all the transformations for the frame are complete (as signalled by the ADVANCE command). Thus the sequence of commands

```
MOVETO(PIC1,X1,Y1);
DRAW(PIC1);
ADVANCE;
```

produces the same results as the sequence

```
DRAW(PIC1);
MOVETO(PIC1,X1,Y1);
ADVANCE;
```

YACAS builds a list of all the pictures it is told to DRAW in a frame; this list is implemented as a threaded list through the picture header table. When the ADVANCE command is issued, this list is traversed and all pictures on it are drawn, then removed from the list.

In addition to this postponement of display, YACAS ensures that only whole pictures appear in the list. YACAS checks each picture it is told to display to see whether the picture is a root or a son. If it is a son, YACAS traces back up the structure to find the root picture and adds this to the list (thus, complete articulated pictures are drawn if any of their parts are specified to be drawn).

Film File Picture "Display"

As mentioned in section III.3.5, the film file contains picture and cel description information only when there has been some change made to the picture or cel description since the previous time it was plotted. The method used to achieve this is as follows: when a picture or cel is created, flag bits in the header are set to indicate that there "has been some change". Any command that makes changes to a picture's header or to a cel description also sets these flags (note that if only a picture header is changed, only the picture's flag is affected; the associated cel's is unaffected). When a picture is to be entered in the film file, the picture header "changed" flag is inspected; if set, the picture header is added to the film file and the "changed" flag is cleared. Similarly, if

the cel's "changed" flag is set, the cel description is added to the film file and the flag cleared. In the case of articulated pictures this process (checking the "changed" flags and copying those pictures and cels that have) is repeated for each picture in the structure.

One further aspect of picture display via the film file concerns the effect of deleting a picture. The display device is assumed to have "total recall" - any picture or cel description it has encountered in the film file is remembered until either it is replaced by the appearance of another description, or it is explicitly deleted by one of the film file "Delete" commands. In the main program data-structures, YACAS maintains a "copied" flag which is used to indicate whether a picture (or cel) has ever been copied to the film file. This flag is set whenever a copy is done; it is reset only when the picture or cel is explicitly deleted via the Delete command.

Paper Plots: Cartooning

The primary uses for paper plots in YACAS lie in the areas of documentation and debugging. The paper plots serve as hard-copy records of the graphic images produced by the user's program.

Since a film of even modest length consists of quite a large number of separate frames (720 frames in a 30 second sequence) some method is needed to reduce the number to be drawn (to see them all the user can inspect the film; to document the film only a few are needed). YACAS selects for display every nth frame, where the value for "n" is determined by the user via the CARTOONING command.

As part of the CARTOONING command YACAS provides the capability to plot 1, 2, 3, or 4 (non-overlapping) viewports across the paper plotter width. The user thus has no control over the absolute vector lengths drawn on the paper plotter (and thus can not draw carefully scaled drawings

using YACAS); what the user does get is a set of properly scaled frames in which everything is proportionately correct. By selecting the number of viewports to be used the scaling can be set to the most suitable size for the amount of detail involved.

IV.2.2 Motions

YACAS can be thought of as operating in two different modes: static and dynamic. In "static" mode, transformations are applied to pictures immediately: a MOVETO command causes the picture header's position co-ordinates to be immediately changed. If the picture is then drawn it will be drawn at the new position. "Dynamic" mode is used to indicate that a transformation is to be applied bit-by-bit over some specified interval. Taking the MOVETO example, the picture header's position co-ordinates are changed by small increments for each frame such that the final values are reached only at the end of the interval.

These transformations applied in the dynamic mode are called "motions" in this thesis.

Motions thus encompass more than simple translation: the SCALEBY and ROTATETO commands may also be "motions"; INTERPOLATE is also a "motion" which involves picture (cel) shape change. It is reasonable to extend the concept of "motion" to deliberate variations of other parameters of a picture. For example, FADE and DISSOLVE are also motions; these affect the picture's intensity. PAN and ZOOM are motions as well, even though they do not apply directly to pictures at all: they affect the size and position of the display window.

Several aspects of the implementation of motions are discused below.

Motion Scheduling

YACAS provides the user with a mechanism to express what motions are to occur, how long they are to take, and when they are to begin. The "time" aspects of this are expressed with the INTERVAL command (for a discussion of what "time" is and the units used to measure it - insofar as YACAS is concerned - see the User Guide section 2.3). This command signals the YACAS system that it is to switch to dynamic mode (if not already in that mode) and indicates the starting time and duration of a set of motions to be described. The motions to be carried out in the interval are specified with separate commands. The association of motion commands with the interval is accomplished through the execution sequence of the user's program.

Most YACAS motion commands operate in both static and dynamic mode. Of those implemented, only INTERPOLATE does not operate in static mode. Motion commands are generally implemented as two distinct subroutines. One routine is the "command" the user uses: it first checks to see whether it has been called in static or dynamic mode (via a boolean variable set by INTERVAL and cleared by ROLLIT). If the mode is static, the code required to carry out the Static operation of the command is executed; if the mode is dynamic, an entry is made in the agenda indicating what routine is to be called, when it is to start and end, and the parameters to be passed to it (see section IV.1.2).

The other subroutine of the pair implementing a motion "command" handles the dynamic operation. This routine is not available to the general user; it is called by ROLLIT when the agenda is being processed. When called, this routine is passed the parameter information placed in the agenda by the static mode routine, the start, stop and current frame values, and a few other items of information to be discussed below. The routine uses the information passed to it, together with the information in the picture

and cel descriptions, to determine the amount of motion to be carried out for the current frame. In some cases the dynamic routine may want to save some information from one invocation to the next. YACAS provides a small amount of "memory" for this purpose in the agenda: two of the four "parameter" entries are available as "read only" (may be set only by the static mode routine and thereafter not altered) while two others are "read-write" (may be altered by the dynamic mode routine). The agenda thus may contain both scheduling information for the commands to be used, and some execution history information for each of the commands.

Incremental Motion

The picture and cel data structures always contain information about the "current" postion, size, shape etc. of each picture and cel. During execution, the dynamic mode routine must determine the amount of change to apply for the frame given the "current" situation as reflected in the data structure, and the "parameter" information from the agenda (this parameter information usually indicates the final values to be achieved).

An example involving a translation motion with linear dynamics is easiest to explain.

Looked at from an over-all perspective, the motion involves subdividing the path from starting position to ending position into a set of equal-sized steps (the number of steps equal the number of frames in the interval); the picture is positioned at each of these points successively to achieve the motion. Viewed incrementally, the problem becomes slightly more tricky: the amount of motion remains constant, but the proportion of the remaining distance increases (if the motion was to take 5 frames, the proportion of the remaining distance from the current position to the final is 1/5 for the first frame, 1/4 for the second, 1/3, 1/2, then finally 1/1 for the last frame).

The arithmetic is straight-forward for the linear dynamics case but gets more complicated for other forms of dynamics. A discussion of the arithmetic involved is presented in section 5 of the User Guide.

Concurrency

A ball rolling from one place to another might be thought of as having one form of motion: the "roll" motion. A more useful way to think of it is as two motions that are acting on the ball simultaneously. One is a translation which moves it to the new place, and the other is a rotation which causes it to rotate about its center. YACAS provides a basic set of motions that may be combined to produce more complicated ones (such as the ROLL). The word "concurrent" in its several forms will be used to indicated the apparently simultaneous application of motions to one or more pictures.

The mechanism for specifying concurrent motions exists as part of the INTERVAL scheme. All motion commands issued subsequent to an INTERVAL and before a ROLLIT are considered to be applied during the same interval - they will act concurrently. Concurrency does not exist in just single intervals; intervals may overlap to any extent with the result that motions in separate intervals may be concurrent also.

YACAS manages concurrency through the agenda. The events scheduled in the agenda are carried out through the ROLLIT command. Naturally they are not processed concurrently; they are processed sequentially in the order they appear in the agenda. They appear to act concurrently because only the result of all the motions for a simple frame are displayed.

Dynamics

The rate at which a motion is carried out can be called the dynamics of the motion. The example above (in section IV.2.2.2) illustrated the effect of "linear" dynamics - a motion takes place at a constant rate over the course of an interval. YACAS also provides two other motion dynamics: "cushioned" dynamics and "sinusoid" dynamics (see User Guide for a description). The association of dynamics with a motion is accomplished by the sequence of command execution - a dynamics command is associated with the most recently executed motion command by modifying the agenda entry for the motion (by adding the dynamics indication).

The dynamics are applied in the following manner: each dynamics scheme has an associated function which returns a value in the (inclusive) range 0 to 1 calculated from the starting, ending and current frame numbers and one parameter that is maintained in the agenda (the SINUSOID function uses this parameter to store the period value). The value calculated represents the proportion of total motion (from beginning to end - not from current position) required for the current frame.

To apply the dynamics, the motion routines are passed both the current dynamics function value and the previous one. The "previous" value is stored in the agenda, set intially to zero. The motion routines must construct from these two values the amount of incremental change required. The problem of working out the amount of incremental change becomes a modest algebraic excercise which will not be examined here; the implemented solution may be found in the source code of the appropriate routines.

IV.2.3 Linear Interpolation

The linear interpolation algorithm has been described elsewhere (eg. [Burtnyk:1971] where it is called "key-frame animation") and has been implemented in several computer animation systems (eg: ARTA, ANTICS, and the Cornell system). As the technique is quite powerful, it will be briefly described below in conjunction with a discussion of its implementation in YACAS. A short discussion of the merits and demerits of the technique may be found in [Levoy:1977].

The linear interpolation technique is an algorithm that can be used to change the shape of one picture into the shape of another. The basic algorithm is (perhaps suprisingly) simple, though it works only with vector graphics systems. Two pictures are required - one representing the starting picture and the other representing the final picture. The vector lists which describe the two pictures must be the same length, and a one-to-one correspondence between points in the same relative positions in the lists is assumed.

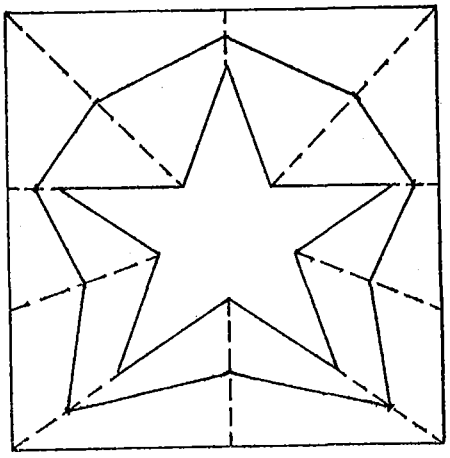


Fig. IV-6
Linear interpolation - a 1/2-way picture.

Each "inbetween" picture is constructed in the following manner (take, as an example, the fifth frame of a ten frame sequence - the 1/2 way picture). The intermediate picture is constructed from a set of points positioned the appropriate proportion of the distance between the corresponding end points in the key frames' vector lists. For the example suggested, the "appropriate" proportion would be 1/2 (see fig IV-6). In the 10-frame sequence, the first intermediate picture would be composed of a set of points 1/10 of the distance between the corresponding key frame points. The next frame would be 2/10 of the distance; the next 3/10, until the last frame would be 10/10 of the distance (the same as the final frame).

A bit of reflection on this process will show that for each individual point in the picture, the method is the same as a MOVE motion where the starting and ending points are known. The difference is that each point in the original picture has its own specific ending position; the method of specifying all the required "ending" points is to use an equal-sized picture of the desired shape.

Two principal problems arise: ensuring that both pictures have the same number of points, and relating the "connectivity" of the first picture to that of the second.

Where the pictures involved are very simple - consisting of a single curve with only a few carefully-chosen points (regular polygons for example) - few difficulties arise. Where the pictures consist of free-hand sketches with a large number of points each, it is generally very difficult to ensure that both pictures have exactly the same number of points as the algorithm demands. Where the pictures have more than one curve, the problems compound. Not only must both pictures have the same number of points in total, but they should also have the same number of curves, and each corresponding curve should have the same number of points. If two pictures are used whose

corresponding curves do not have equal numbers of points (even though the pictures as a whole do), then in constructing the intermediate pictures, a decision about connectivity must be made for at least one of the intermediate pictures (which connectivity is to be used?). If the connectivity of the initial picture is used throughout the interpolation, then the final picture will not look right; otherwise a decision must be made about the stage at which the connectivity change is to take place.

Solutions to these problems are all rather ad hoc. ARTA used the following schemes. The connectivity problem was given one solution: the place where the connectivity of the intermediate picture changes from that of the starting picture's to that of the final picture, is at the 1/2-way point in the sequence. This can cause the abrupt appearance and disappearance of vectors at the mid-point of a sequence, but is easily implemented, simple to understand, and the difficulties can be avoided when understood. The "number of points" problem was solved in two ways. The simplest solution involves the algorithmic addition of points (spaced evenly along the existing vectors) to the picture containing the fewest points thus forcing both pictures to have the same number. Adding the new points along existing vectors ensures that the shape is preserved. The second scheme adds points on a curve-by-curve basis so that both pictures have the same number of curves (adding curves as necessary) and the same number of points in each curve; this usually results in changes to both the starting and final pictures.

Implementing Interpolation in YACAS

Interpolation in YACAS can only occur within a single interval. Moreover interpolation affects only the cels of the pictures involved. Interpolation is thus used solely to effect a change in the shape of a picture. If the picture whose shape is to be changed is part of an articulated picture, only the cel of the specified picture changes; not

the cels of all the pictures in the group.

Another difference between interpolation as a motion and other motions (such as MOVETO) is that the pictures specified in the command are necessary only to provide an indication of the cels to be used. As a result, the picture whose cel is being changed is not implicitly drawn by the "system". If the user wishes to change the shape of a picture and draw it as part of an interval, then two commands must be given: the INTERPOLATE command and the DRAW command. The reason for this apparent anomaly is rather subtle. ROLLIT was originally intended to draw pictures that were changed in an interval. INTERPOLATE changes only cels; the mention of pictures is necessary only to indicate the cels involved in the interpolation.

In YACAS, the INTERPOLATE command "solves" the problems of connectivity and equalization of point counts in a manner analogous to the first technique indicated above. Connectivity is changed at the half-way step of an interval, and necessary extra points are added evenly along the total existing vector length. The generation of intermediate cels is accomplished by changing the points in the cel table (the cel is not deleted and replaced with the new one; the new point values replace the previous ones). The method of incremental change described above is used to calculate the positions of the intermediate points. This implementation is quite simple; its main drawback is its total disregard of the curves making up the cels. Further developments to YACAS should include refinements to the interpolation scheme as a high priority.

IV.3 Division of Labour

In performing its various functions, YACAS must perform a number of arithmetic calculations. Certain of these calculations are performed as part of the user's main

animation program, while others are performed by the film previewing program, PLABAK. The following discussion describes which calculations are performed in each component of YACAS.

IV.3.1 Main Animation Programs

In an animation system, two principle forms of calculation need to be performed: calculations that determine which pictures are needed in each frame and what they should look like, and calculations that transform the pictures from internal format to display-file format. Most of the commands the user calls perform calculations of the first sort; YACAS "system" routines perform the calculations of the second sort.

In the user's main program the YACAS system does all the calculations concerned with maintaining the data structure. Of the commands implemented, only INTERPOLATE involves much calculation; the others (MOVETO, SCALEBY, etc) involve only small amounts affecting the picture's header.

If the paper plotter is selected as an output device, additional calculations are required to select the correct frames to be drawn, and to do the transformation of picture data from internal form to the form required by the plotter. These latter calculations involve the application of the picture position, scale factors and orientation attributes to the cel vectors, clipping these vectors against the display window, and finally mapping them into the viewport co-ordinates.

When "FILM" has been selected for output, the YACAS routines do not do any of the data-base to display-representation calculations. All they need to do is maintain flags which indicate which pictures have been copied to the display file, and which have been recently

modified. In addition, to actually write the film file, calculations are performed to transform numeric values from the Burroughs format to PDP-11 format.

IV.3.2 PLABAK Calculations

The PLABAK program's primary task is to transform the picture data-base vectors to display-file form. It does this by reading the film file and carrying out the "commands" found there. Its primary calculations are the same as those for the paper plotter. The rest of its work concerns maintaining its data-base and synchronizing the display file calculations with the display of those files. PLABAK maintains two display files: one that is displaying the "current" frame, and one that is being filled with the next frame's image.

Both PLABAK and the paper plotter handling routines do their calculations on a vector-by-vector basis. This eliminates the need to store intermediate forms of pictures or curves.

Chapter V
The User Interface

The previous chapters have described the design and some of the implementation strategies employed in the YACAS animation system. This chapter will discuss the way the system can be used to make films. The details required to get an actual film file produced and displayed on the GT44 are left to the User Guide; what is included here are the over-all ideas and methods involved.

V.1 Using the YACAS Subroutines

To make a film one must write a program. This program calls the YACAS subroutines in a sequence that describes the pictures to be used and the film sequences to be produced. The program produces a film file which may be displayed on the GT44 and filmed from that display.

The subroutines the user calls are written in ALGOL. Under the current implementation, the user appends these routines to his source program so that they are compiled together. The more common approach of providing a pre-compiled set of routines to be linked into the user's program (Burroughs uses the term "binding") has not been used because it was found to be too costly (an experiment involving three programs prepared using YACAS in both forms produced costs about 10% higher for "compile and go" execution of the form using separately compiled subroutines; compile costs were much lower but binding costs, particularly the I/O charge, were much higher). This approach has two principal consequences (plus a number of lesser ones): users of languages other than ALGOL are precluded from using the system, and sophisticated users of the system have a great deal of "power" to modify and extend

the system to suit their particular needs or wishes.

The restriction that the system may be used only by ALGOL programs is a serious problem, but one which can be solved by revising the implementation slightly. A small number of changes to the YACAS source file can allow it to be compiled into separate modules that can be bound to FORTRAN host programs. No features of the ALGOL compiler have been used in the YACAS subroutines that would cause incompatabilities between the two program elements. Certain YACAS "commands" would need to be re-written as subroutines - those that are currently implemented as ALGOL "DEFINE" statements.

For the ALGOL programmer, a single "INCLUDE" statement will include the text of the YACAS source routines (with the "LIST" switch reset at the beginning and "POP"ed at the end).

The user's main program may call YACAS subroutines in any suitable order. Certain sequences of commands are "syntactically" incorrect and will, in most cases, be flagged by the system. There are only a very few sequences that must be followed.

The very first YACAS routine the user calls should be the "ENVIRONMENT" command. This initializes the YACAS data base, and, via its argument, indicates which output "devices" are to be used. Output devices are "FILM" for the film file, "PAPER" for the paper plotter, "BOTH" for both film file and paper plotter output, or "NONE" (anything else is treated as "NONE" - producing no output can be useful when debugging programs). Failure to call the ENVIRONMENT routine will result in unpredictable results.

The only other subroutine that must be called is the "FINISHED" routine which empties the output buffers and closes the output files. It also prints a message on the YACAS printer log indicating the number of frames produced

and the size of the film file (if one was opened).

Two other sequences that must be observed may be used an arbitrary number of times between the ENVIRONMENT and FINISHED calls. One sequence is used to add pictures to the YACAS data base, while the other creates film sequences.

To add a picture to the YACAS database, the picture description begins with a call to OPENP, is followed by calls to VECT, ARC, or POLYGN to describe picture sections on a vector by vector basis, with calls to BGNOUT, and BGNATT to signal the beginning of outline and attachment point sections respectively. Whole pictures can be defined using the READCEL or USECEL subroutines which create appropriate cel descriptions either by reading the vector coordinates from cards (READCEL) or by indicating that an already-existing cel description is to be used rather than a new one. When a picture description is complete, the picture definition is completed with the CLOSEP command.

When a new picture is created (via the OPENP routine) a number (a positive integer less than 16384 in value) is returned. This is the "picture number" and is the means by which pictures are identified by YACAS. Until a picture is explicitly DELETED, it retains the picture number it was originally assigned.

The description of an articulated picture is accomplished via a slight variation on the above sequence. When a picture is to have "sons", it is opened and described as indicated above. Before it is closed, however, its sons are described by opening, describing and closing them. When all sons have been described, their father is closed. To indicate the attachment point that a son is connected to, the ATTACHTO command may be used before the son is closed. An alternative method of defining an articulated picture is to add a pre-existing picture to the currently-opened picture as a son using the ADDPIC command; the ADDPIC

command must be used after the currently-open picture has been completely described, and before it has been closed.

To create film sequences, two orderings of YACAS commands must be adhered to:
(1) A film sequence begins with the execution of an INTERVAL command and is followed by one or more motion commands. An interval command remains in effect until either another INTERVAL command, or one of the SETSTATIC, or ROLLIT commands is executed.
(2) Within an interval, to specify dynamics for a motion the desired dynamics command must be executed immediately after the motion command it applies to. Dynamics commands have no meaning in any other context.

V.2 Using PLABAK

PLABAK is a program to be run on the GT44 to perform the function of a previewing system. Detailed operating instructions may be found in section 4 of the User Guide. The following discussion indicates the uses of the program in more general terms.

PLABAK may be used to display the contents of a film file on the GT44. It attempts to do so at the full speed of the film (actually 25 frames per second rather than 24) and displays a warning message when it is unable to do so because of excessive computational load. The user may stop the preview of a film at any time to examine the current frame then continue the displays at full speed. When the preview is stopped, the user may issue any of the commands (described in the User Guide). PLABAK previews a film in the forward direction only.

PLABAK commands are issued by responding to the PLABAK prompt symbol ("*") with an RT-11 "switch" with one or more values. A switch is a single alphabetic character preceeded by a slash ("/" -ie: /T); a value is a number. The number

is treated as octal when preceeded by a colon (":"), or as decimal when preceeded by an exclamation mark ("!"). For example, /X:11 is the same as /X!9.

The user has control over two aspects of the film display rate: the length of time (in clock "ticks" (1/50 sec)) each frame is to be displayed (the /X switch) and the frame selection criterion (the /S switch). The frame display time (/X) may have any value between 0 and 32767 (default is 2); 0 indicates that PLABAK is to wait after each displayed frame for the user to type something (usually the return key) before going on to the next frame. With the /S switch the user can indicate the number of film frames to be skipped between displayed frames (negative values of /S) or the number of frame display times each frame is to be displayed (positive values of /S).

Other controls available to the user include the ability (a) to specify sub-sequences to be displayed (the /B switch) and (b) to override the ZOOM and PAN commands in a film and to display the film through a fixed window specified during PLABAK (the film's window appears as a dashed rectangle).

The technique used to transfer film files from the B6700 to the GT44 is described in the User Guide. This technique relys on several special-purpose programs which require files in a format slightly different from that required by PLABAK. The transfered file may be re-formatted to a form acceptable by PLABAK by the TBSCAN program. TBSCAN has the alternate use of producing a command-by-command listing of a film file that can be useful as a debugging tool. Its use is described in the User Guide.

V.3 A Simple Language - FILMMAKER

In some situations, a film may be able to be made which does not need any of the features of the host language, and uses the host solely to contain the calls on the YACAS subroutines. Educational films and some forms of "art" film can often be made this way. To facilitate this form of use, a simple "language" has been devised which allows the user to make a film using YACAS commands "directly". This language is called FILMMAKER.

FILMMAKER is, quite simply, a program which reads input records from a file (usually a card file), treats each record as a simple YACAS command and calls the appropriate YACAS subroutine using the parameters supplied in the record. The implementation of this "language" is centered around use of the Burrorghs ALGOL "Free-Field" READ feature.

A few, very simple rules apply:

(1) each input record (card) may contain only one command.

(2) each command consists of the command name (which may be abbreviated to the first 6 characters) followed by the necessary parameters (in the proper order), optionally terminated with a semi-colon followed by a comment. The command name and following parameters are separated from each other by commas.

(3) a command must fit entirely on one input record.

(4) an input record may contain no command; it will be treated as a comment.

(5) an input record containing an unrecognized command will be ignored; surplus parameters will be ignored; too few will cause unpredictable results.

(6) where pictures are used, the user identifies the picture by number (in the range 1 to 50 - in contrast to the normal YACAS approach by which YACAS assigns picture numbers).

FILMMAKER is quite limited in what it can do. The FILMMAKER commands are processed in a straight sequential

manner. No branching or looping is available; no arithmetic operations are available, nor are any variables. FILMMAKER provides no mechanism by which new commands can be added to the supplied set. Details concerning the use and limitations of the FILMMAKER program may be found in the user guide.

FILMMAKER's purpose, and major advantage, is that it can be used to efficiently produce simple "one-off" films. Such films may require slightly more planning than films produced in the "normal" way, but are generally more quickly processed and less expensive to produce since no program compilation is required.

Chapter VI

Conclusions

VI.1 The Grand Design

The previous chapters have described YACAS as it has been implemented. YACAS in this form is rather restricted in what it can do. The concept of YACAS in the beginning was for a much more versatile batch animation system which is described below.

The grand design includes the following features in addition to those already mentioned.

(1) A subsystem providing for the preparation of free-hand sketched pictures prepared on the GT44.

(2) A picture library scheme where-by pictures can be stored on disk or tape for subsequent use by animation programs.

(3) Improvements to the picture quality such as the ability to produce coloured pictures with surface shading and hidden lines removed.

(4) A broader range of motion commands, including commands for path-following.

(5) The provision of "text" pictures to provide titling and labelling capabilities. The text would be provided in a variety of type fonts and would be able to be manipulated as any other picture.

(6) A library scheme would be provided for film files to assist in keeping track of the various scenes and versions of the films produced.

(7) A film editing subsystem would be provided to allow film files to be flexibly combined.

(8) The film preview program would be made more effectively interactive and would have enhanced capabilities.

VI.1.1 Division of Function

The complete YACAS system planned to consist of four distinct subsystems corresponding to the functional subdivisions of (a) picture preparation and archiving, (b) film production, (c) film file editing, and (d) film display and recording.

Picture Preparation and Archiving

One of the critical areas of an animation system that requires close attention is of picture preparation. The picture preparation aspects of YACAS as currently implemented - the input of pictures from data in a card-file (originally, encoded by hand), or the description of pictures by program statements - is inadequate for all but the simplest of films. "Real" pictures, whether simple or articulated, are usually much more complex than can conveniently be prepared with the existing facilities. As indicated above, two additional features are part of the broader design of YACAS to carry out the picture preparation function.

The basic component is a separate subsystem to allow pictures to be prepared as free-hand sketches. Such a system requires specialized hardware, and the only available hardware is the GT44. The main part of the picture preparation subsystem would thus be a program allowing the user to prepare free-hand sketches on the GT44 using the light pen, a tracking object, and the display screen for data input and display. The system would be as interactive as the hardware allows - relying mainly on menus, light-buttons and gestures with the lightpen for input, with visual feedback on the display. A limitation of this scheme is that the digitizing of prepared material, or the copying of pictures from books, periodicals etc, would be difficult.

Associated with the picture preparation program would be two "picture libraries", one on the GT44 for the storage of picture descriptions as they are prepared, and one on the B6700 for the storage of the pictures pending their use by an animation program. There would, in fact, be several picture libraries on each computer since users of the system would want their own private libraries. Associated with the libraries would be utility programs to transfer pictures between libraries, delete pictures from libraries, list library contents, and plot pictures from the libraries. Additional commands would be added to the set of YACAS commands to allow animation programs to obtain pictures from the libraries and to store new pictures into them.

Film Production

The film production function of an augmented YACAS system would include extra features, some of which have been alluded to previously.

(1) Three picture quality attributes would be added: the ability to colour pictures, to shade areas of pictures, and to remove hidden lines where one opaque picture covers part of another. An approach to the implementation of these features is discussed below in VI.2.

(2) A new picture type - the "text" picture - would be introduced. In a text picture the detail section of a cel would contain a character string, treated as a picture, that could be used for titling, labels, and explanatory comments in films. A variety of character fonts, such as those designed by Hershey [Hershey:1967], would be provided. Comments on the implementation approach appear below.

(3) A variety of new motion commands would be added. Some would be variations on existing ones (MOVEBY, ROTATEBY, SCALETO) while others would be completely new (eg a REVEAL command whose action would be the successive disclosure of parts of a picture from nothing to the whole picture over a stated interval). Some of the new commands would operate on

pictures (such as those just mentioned) while others would affect the displayed frame globally - new "camera" motions such as wipes and masks. The exact number and nature of new motion commands is not yet specified, and would best await some feedback from the use of the existing system.
(4) Path following would be added as a technique for specifying motion. Current plans do not involve use of path-following with the generality suggested by Baecker in his P-curves, but only as a means of specifying picture position. The author feels that the use of paths for controlling other attributes is thought to be inappropriate for a batch animation system - the time delay between sketching a path and seeing its effect would negate the value of the technique's flexibility).

With any software system, it is easy to think of improvements that can be made, some "large" and some small. The above items were considered for inclusion in YACAS but there was not enough time to implement them. With a small amount of reflection one can think of additional enhancements, and practical use would yeild many more.

Film File Editing

Long films are rarely produced complete; they are usually produced as a set of separate sequences. These sequences, at some stage, must be combined to produce the complete film. This may be done at the film stage using conventional film editing techniques, but additional skills and equipment are needed to do so. Since the sequences exist in film file form anyway, it seems natural to provide a method by which the film files may be combined using the computer. A film file editor will do this.

Tilson has described a film editor with a great variety of sophisticated and useful features [Tilson:1975]. Besides the ability to combine separate film files, his system allows the user to manipulate the film in a variety of ways -

chiefly the manipulation of the timing of the film. Although the features of Tilson's editor would be useful, the editor intended for YACAS would be much less elaborate; rather closer to that proposed in [Britton:1973].

The major features of the YACAS editor would be
(1) The ability to "cut and splice" - to simply concatenate film files to create longer ones.
(2) The ability to extract parts of film files for concatenation.
(3) Manipulation of timing through frame selection techniques equivalent to those provided in PLABAK.
(4) The ability to overlap sequences or parts of sequences giving "double exposure" effects.
(5) The ability to control the film image intensity to allow sequences to be made brighter or darker, to introduce fades, and (where sequences overlap) dissolves.

The film file editor would be implemented on the GT44. It could be done on the Burroughs where more file space is available, but on the GT44 it would be able to be used in conjunction with PLABAK in an interactive manner. Film sequences could be edited together, then immediately previewed and any changes needed applied at once.

Film Display and Recording

PLABAK provides a preliminary approach to the display of film files. It embodies many of the major features envisioned for it, though in a rather crude way. Among the features an enhanced version of PLABAK would provide are
(1) A more pleasing form of interaction. Greater use would be made of menus and the light pen for the user to control the program.
(2) Film sequences would be able to be previewed in both forward and reverse directions.
(3) Better control over the speed of playback would be provided, particularly the ability to preview film material

at a proper real-time rate.
(4) With the additional image quality features of YACAS, PLABAK would be able to display shaded, hidden-line-removed pictures, and text strings. The level of detail displayed would be selectable by the user - whether to actually shade areas for a particular preview sequence, whether to remove hidden lines, whether to use the hardware character generator rather than transliterate text by software to the proper character font.
(5) With a small amount of hardware development to enable the GT44 to control a movie camera (some way to trip the camera's shutter), PLABAK would be able to record films automatically.

Film File Display

These enhancements would introduce a very heavy computational load on the GT44. Such a load would probably mean that the GT44 would be unable to display films at the desired rate. This problem would be over-come by making film display a two-pass operation. The first pass would construct a sequence of display files, one for each frame of the film, without displaying any of them. The second pass would be responsible for their display at the correct rate. Built into the first pass would be an algorithm to determine whether a frame contained too much information to allow the frame to be displayed in one frame time, and program code to simplify the image until it could. Simplification algorithms such as those suggested by [Tilson:1975] could serve as a model.

VI.1.2 Implementation

The following is a brief discussion of the way several aspects of the enhanced YACAS would be implemented. The particular items discussed deal with image enhancements. These were chosen to indicate the reason for the "outline section" included in the YACAS cel descriptions and to show

how this outline was intended to be used.

Surface Shading

With a vector display system, a common way to fill in an area representing a surface is to draw a number of closely spaced vectors through the area. Usually these are a set of parallel vectors spaced close enough that the vectors blend together to form an apparent solid. The vector spacing required to produce solid shading depends on a variety of factors - the resolution of the display, the width of the displayed vectors, their intensity, and, sometimes, their orientation.

Functionally, shading would be implemented in PLABAK. When previewing or recording a film the user would be able to specify whether surface shading was wanted, and would be able to select one of several shading densities to be used (controlling the spacing of the shading vectors). Details such as the orientation and spacing of shading vectors for pre-defined densities would be determined empirically.

An algorithm used for surface shading is described by [Dwyer:1967]. In essence, the procedure is to pass an infinitely long trace vector through the picture to be shaded (the outline section of a picture's cel). A list of points where this vector intersects the outline is created, and solid line segments are produced joining pairs of points in this list (see fig VI-1). The trace vector is shifted from one side of the outline to the other by an amount equal to the shading vector separation. The outline must consist of one or more closed polygons; "holes" in the outline can be produced by having closed polygons inside other closed polygons (as in fig VI-1). The order of the points defining the outline polygons, and the shapes of these polygons, are not important since the list of intersection points is sorted before the shading vectors are produced.

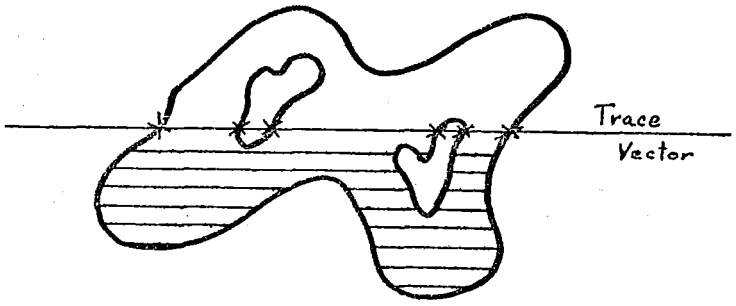


Fig. VI-1
Trace vector intersecting an outline.

Hidden-line Removal

The basic scheme for hidden-line removal would be based on a "2 1/2-D" technique. Each picture would have associated with it a "level" number indicating a depth ordering of the pictures. To simplify matters, a limited set of levels would be provided. The outline sections of pictures would be used to indicate which parts of the picture are opaque.

Starting with the level at the greatest depth, all pictures at that level would be drawn into an intermediate picture area. Pictures on succeeding levels would be added by first removing from the intermediate picture any vectors "hidden" by the opaque parts of the pictures being added. In fact this process would be a two-step one - first the hidden vectors would be removed by comparing the intermediate picture with all the pictures on the new level, then the new pictures would be added to the intermediate one. The two steps are necessary to ensure that pictures on the same level do not hide each other. The level-by-level addition of detail would continue until all pictures had been added to the intermediate one, which would then be displayed.

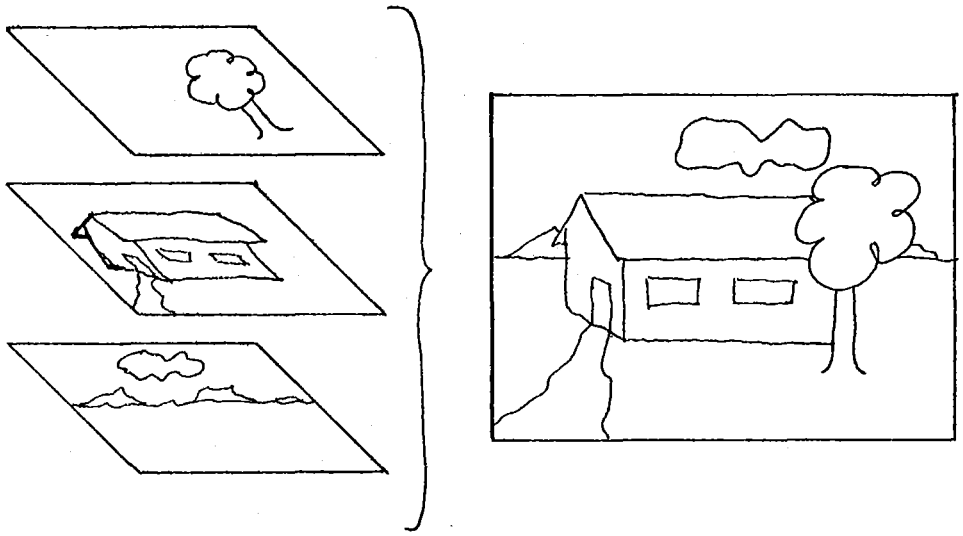


Fig. VI-2
Pictures on "levels" - 2 1/2-D
hidden line removal.

The term "2 1/2-D" is suggested by the fact that the individual pictures are all 2-D images while the levels provide a partial 3rd dimension.

Colour

The production of coloured images by an enhanced YACAS system would involve the production of separate sequences of film for each of the specific colours required. These sequences would then be combined in a film lab through appropriate colour filters to obtain the coloured final film. The user would make coloured films by associating with each picture a number representing a specific colour. PLABAK, when recording the film for colour printing, would separate each frame into separate sub-components based on colour and produce the appropriate separate sequences. Colour could be added by this technique to either the "wire-frame" images currently produced by YACAS, or to the surface-shaded ones.

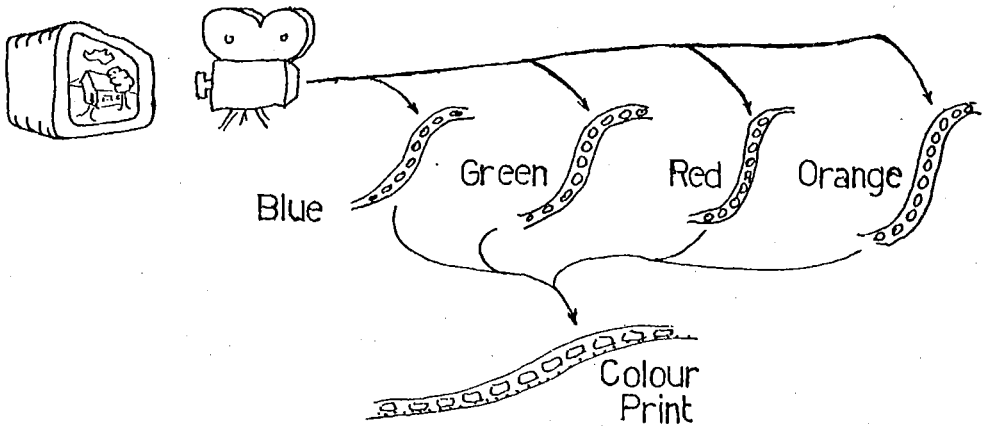


Fig. VI-4
Colour Separation.

A number of problems would require further investigation to be resolved.

(1) First, and foremost, it would be necessary to find a processing laboratory prepared to do the necessary optical work.

(2) Where different-coloured portions of separate pictures overlap, undesirable combination of the colours can result. [Burtnyk:1973] discusses this problem, and proposes solutions based on the preparation and use of "travelling matts" (opague images on film used when printing a negative to mask out areas that are not to be exposed). The use of travelling matts requires even closer co-operation with the film lab (and costs a great deal more). The problem can be solved by software - it is really only a variation on the hidden line problem.

Colour added by this approach would not be "full" colour - coloured films produced this way would have only a small number of discrete colours and subtle colour shadings would difficult to provide.

Other techniques for adding colour do exist. The "Technicolour" process used in the 1930's for live-action films (where three B&W negatives - one for each primary colour - are printed together) is used in the ANTICS system

to achieve full colour reproduction. The coarsness and poor stability of the intensity control on the GT44 rules out its use here.

Text Pictures

Text pictures would be implemented as a new type of cel. The "curves" of the detail section of the cel would contain a character string, the co-ordinates of the string's starting point (lower-left corner), the character height, orientation of the string, and a font code. The strings would be stored in the appropriate character format (EBCDIC on the B6700, ASCII on the GT44) rather than as vector lists.

PLABAK would translate the strings to the appropriate vector lists by looking up each character in a table for the particular font. To allow text pictures to be manipulated like ordinary pictures (in particular, to allow for interpolation from one string to another) a command would be added to the set of YACAS commands to translate text pictures to vector pictures.

VI.2 Criticisms and Enhancements

The major difficulty is that YACAS has not been used by anyone other than the author, and then only to the extent necessary to indicate that it does what it is meant to do. No specific films have been made other than very short, experimental ones, and no films have actually been recorded. Thus it has not been possible to base these criticisms on the experience of actual use of the system by a variety of people.

There are three major shortcomings to YACAS.
(1) The present implementation requires the user to include the YACAS routines in his program as additional source statements. This means that (a) the YACAS routines are re-compiled each time they are used, and (b) only programs

written in ALGOL may use the YACAS subroutines. As noted in Chapter V, the cost of re-compiling was less than the cost of binding the pre-compiled routines into a user program so that the real criticism is of the inefficiency of the ALGOL binding process rather than of YACAS itself. The availability of FILMMAKER is an attempt to reduce this cost for simple films. On the other hand, the fact that programs in other languages such as FORTRAN cannot use the YACAS routines is a severe limitation. A slight re-structuring of YACAS so that pre-compiled routines would be available for binding is possible; the sections on using and extending YACAS would then need to be re-written.

Retaining the current implementation strategy, the cost of using YACAS could be reduced by supplying subsets of the system (YACAS without INTERPOLATE or without articulated pictures for example), thereby reducing the number of YACAS source statements to be compiled. This could be done via the "user $-option" facility of the ALGOL compiler (already used to omit the paper plotter supporting routines via the NOPLOT option - see User Guide Appendix 1), or by providing separate source files for each subset. Were YACAS to be heavily used, such a scheme could be desirable.

(2) YACAS totally ignores one of the current "problem" areas of computer animation: the smooth combination of individual motions into "compound" motions. YACAS, as with most computer animation systems, treats individual motions as being totally independent of each other. Dynamics functions are provided to allow the rate at which motions are effected to be controlled, but these affect only individual motions. Complicated motions are catered for through the ability to specify concurrent motions, and one technique for having the motions of some pictures depend on those of others is provided through the articulated picture scheme.

Where the individual motions are related, the sequence may be called a "compound" motion. As an example, consider the situation where a picture - PIC - is to be moved from

some position - A - to a new position, B, then from B to C, then finally from C to D. The set of three moves from A to, ultimately, D is to be viewed as a single move but must, for whatever reason, be specified as the three separate ones Without very careful planning, there will very likely be discontinuities in the rate of motion at points B and C as the description of the motion changes from one individual motion to another. The reason for the discontinuity lies in the fact that, in general, the amount of incremental motion in each of the individual motions will be different unless careful planning has gone into the locations of the points A, B, C and D and the time allowed to carry out the motions. The problem is particularly acute where motions are produced via linear interpolation.

A possibly fruitful area for future research would lie in this area. What is needed is a method by which compound motions may be specified by the user, with the "system" (YACAS in this case) treating them as simple motions. Dynamics (such as CUSHIONED dynamics) should be able to applied to the compound motion.

(3) Another area that has received little attention in YACAS - in common with many other animation systems - and which may prove to be a fruitful area for further investigation, is the specification of animation "cycles". In cel animation, a "cycle" is a sequence of cel changes that can be repeated to produce a continuous motion. A common application is a walking cycle for a character (consisting of leg and arm positions for one stride), but other cycles could show twinkling stars, falling rain, waves lapping a shoreline. Within YACAS some types of cycles may be specified using the SINUSOID dynamics (a swinging pendulum for instance). For more complicated cases (coumpound or concurrent motions) the user can usually devise a means of programming the cycle using host language constructions (FOR loops for instance). The provision of some way to specify cycles as a system construct could simplify the user's

programming task.

A number of other smaller criticisms may be levied at YACAS.

(1) The current implementation uses fixed-size data structures (the picture and cel tables, and agenda). The fixed size may mean that more complicated films cannot be made due to lack of room in them. To increase their size could make for unnecessarily large programs in a number of cases (less complicated films). These structures would be better implemented as variable-size arrays through judicious use of the ALGOL RESIZE function.

(2) The current implementation of film time as a discrete variable can be a hindrance. With relatively minor modifications, "time" could be specified as being continuous (whole numbers would still represent "frames", but fractional numbers would be allowed) with the result that a film could be viewed as a continuous process, frozen at discrete instants to provide the images to be displayed. The user could then be provided with a more flexible way of specifying his time units so that he could deal with units more appropriate to his application.

(3) A slightly more serious drawback of YACAS is that, where a motion command controls more than one degree of freedom (a MOVETO, for example, controls motion in two directions: the X and Y), only one form of dynamics can be used to control the rate of the motion. Separate dynamics for each controlled degree of freedom can be useful (the motion of a flipping coin can be produced by changing the size of a circle between 0 and 1 in one dimension while holding the other at 1). The syntax for specifying dynamics and the data structures used in YACAS would make dynamics for multiple degrees of freedom difficult to add. Experience would be the best way to assess the need for modification to the present scheme.

(4) The form of interpolation currently available in YACAS is the simplest of the forms to implement. It serves its

purpose in allowing interpolation as a technique for specifying motion, and in showing how interpolation can fit into the YACAS approach to animation. Other variations on the scheme should be investigated. A particular variation that should receive early attention is one which would match the beginning and ending pictures on a curve-by-curve basis, rather than the point-by-point approach used here.

(5) An annoying clumsiness of YACAS became apparent while preparing the example presented as Appendix B of this thesis. YACAS automatically draws each picture mentioned in a motion command, but only in the interval for which the motion was specified. If a picture is intended to be visible throughout a longer sequence, but moves during only part of it, it must be explicitily drawn (using the DRAW command) for those intervals when it is at rest. Remembering to do so when there are a large number of pictures can be troublesome. A useful addition to YACAS might be the specification of a "background" picture to be automatically drawn in each frame of any sequence. This background picture would in fact be a list of pictures rather than a single one. The user would be provided with commands to add or remove individual pictures from the background, to clear the background altogether (as in preparing for a "cut" from one scene to another) or to specify that "all" pictures were to be part of the background.

Finally, there are several more areas where enhancements may be made. The utility of modifications of this sort is not established but could bear further investigation.

(1) The agenda could be re-implemted as a linked list with new entries added at the appropriate point in starting time sequence. Although such a change could be viewed as just another way to store the agenda, sorting of the agenda would not be necessary, and the next enhancement ((2) below) could be added relatively easily.

(2) Under the present implementation, entries may be made in the agenda only as part of an interval description. It may be useful to allow the execution of a motion command to schedule additional motions, perhaps for a future time. As an example, a BLINK command (which would cause a picture to appear and dissappear regularly) could be implemented as a routine to simply display a picture. It would be called for a frame, draw its picture, then re-schedule itself for the next frame in which its picture is to appear. It would never have to decide whether a picture was "on" or "off" since it would only be called when its picture was "on".
(3) It is sometimes advantageous to allow multiple "cameras" to be simultaneously viewing the data base. YACAS has a single "camera" - the display window. Multiple cameras can be used to give split screen effects similar to those occasionally used in conventional film-making and advertising. Controls would be needed for both the separate cameras and the areas into which these camera images would be displayed.

VI.3 Implementation Lessons

Apart from specific details about program construction, large system design and thesis writing, three general comments might form the basic lessons learned from this project. They suggest ways that this project would be changed were it to be done again.

Firstly, ALGOL would not be used as the implementation language; FORTRAN would be used in its place. This may seem a retrograde step, and may in fact not accomplish its purpose - to provide a less costly animation system. ALGOL is an excellent programming language, and the reasons for its choice are still valid; particularly its program structuring features. FORTRAN is a retrograde step, particularly with regard to program structuring features. But FORTRAN is designed to work effectively with separately

compiled subroutines and it is thought that the binding of subroutines to a user's FORTRAN program would be less costly than either binding ALGOL subroutines, or re-compiling the ALGOL routines as is now done. Further work is needed to see which approach is best.

Secondly, the choice of MACRO as the implementation language for PLABAK would be reviewed. FORTRAN, with some critical routines written in MACRO, might be the best approach. MACRO was chosen to minimise the use of memory and to provide fast execution. Its main drawback is that it took a very long time to write PLABAK using MACRO. It is felt that FORTRAN could have been used to write PLABAK in a much shorter time with relatively small penalties in core use and execution speed. Again, further investigation is needed.

Finally, despite attempts to simplify the project, it was still too large and complicated for the time available. Either of the two major areas tackled - the set of subroutines or the PLABAK system - would have provided ample scope for investigation and development. Had only the subroutines been attempted, a better set of motion commands could have been provided, a more thorough investigation of the costs of binding versus re-compiling the routines could have been undertaken, and the use of the routines from FORTRAN programs could have been attempted. But no films could have been displayed since there was no PLABAK program. Had PLABAK alone been attempted, user interaction with the program could have been made more effective, better control of the film display rate could have been provided, and some way to automatically record the film material could have been devised. But there would have been no films to look at.

As a compromise between time and dreams, YACAS works reasonably well. For others wishing to do development work in the field of computer animation, some of the ground-work

has now been done and more attention can be paid to the details and problems requiring investigation.

Bibliography

The following abbreviations are used in this bibliography to denote the proceedings and journals indicated:

UAIDE - Proceedings of the annual meeting of the Users of Automated Information Display Equipement (the Stromberg-Datagraphics user's group - now part of the National Microfilm Association).

SIGGRAPH - Proceedings of the annual conference on Computer Graphics and Interactive Techniques. Published by the ACM, New York.

IFIP - Proceedings of the IFIP Congress. Published by North-Holland Publishing Co., Amsterdam.

FJCC - Proceedings of the Fall Joint Computer Conference.

SJCC - Proceedings of the Spring Joint Computer Conference.

NCC - Proceedings of the National Computer Conference. Above three published by AFIPS Press, Montvale, N.J.

CACM - Communications of the Association for Computing Machinery.

In addition to books and periodical articles, several entries in this bibliography are for films. A good list of computer animated films and sources (though the list is incomplete), may be found in [Speer:1974].

[Baecker:1969]
R.M. Baecker, "Interactive Computer-Mediated Animation", Thesis, PhD., Massachusetts Institute of Technology,

1969, Project MAC-TR-61.

[Baecker:1970]
R.M. Baecker, L. Smith, E. Martin, "GENESYS, An Interactive Computer-Mediated Animation System", 17 min colour, sound film, M.I.T. Lincoln Laboratory, Lexington, Massachusetts, 1970.

[Baecker:1973]
R.M. Baecker, "Computer Animation As an Aid in Visualizing Complex Processes", Canadian Datasystems, Vol. 5, #3, (Feb. 1973).

[Baecker:1974]
R.M. Baecker, "GENESYS: Computer-Mediated Animation", in [Halas:1974], pg. 97-116.

[Baecker:1976]
R.M. Baecker, "A Conversational Extensible System for the Animation of Shaded Images", Computer Graphics, Vol 10, #2, (SIGGRAPH 1976).

[Baker:1972]
L. Baker, E. Tucker, D. Buckner, "Computer Generated Optical Sound Tracks", FJCC 1972.

[Ballam:1976]
A. Ballam, "Animated Antics and Cartoon Capers", (British) Computer Weekly, Feb 14, 1976, pg. 14.

[Behler:1969]
B. Behler, E.E. Zajac, "A Generalized Window-Sheild Routine", UAIDE 1969.

[Bergman:1977]
S. Bergman, A. Kaufman, "Association of Graphic Images and Dynamic Attributes", Computer Graphics, Vol 11, #2, (SIGGRAPH 1977).

[Blinn:1976]
J. Blinn, M. Newell, "Texture and Reflection in

Computer Generated Images", CACM, Vol 19, #10, (Oct. 1976).

[Blinn:1977]
J. Blinn, "A Homogeneous Formulation for Lines in 3-space", Computer Graphics, Vol 11 #2, (SIGGRAPH 1977).

[Britton:1973]
T. Britton, "Editing Computer Animated Film by Computer", Proceedings of the 3rd Man-Computer Communications Seminar, National Research Council of Canada, Ottawa, May 1973.

[Bronowski:1973]
J. Bronowski, "The Ascent of Man", BBC Television series; book: BBC Publications, London, 1973.

[Burtnyk:1971]
N. Burtnyk, M. Wein, "A Computer Animation System for the Animator", UAIDE 1971.

[Burtnyk:1973]
N. Burtnyk, M. Wein, "Image Quality Considerations in Computer Animation", Proceedings of the 3rd Man-Computer Communications Seminar, National Research Council of Canada, Ottawa, May 1973.

[Burtnyk:1974]
N. Burtnyk, M. Wein, "Towards a Computer Animation Production Tool", Proceedings of the Eurocomp Conference, May 1974, Pub. Online Ltd, Brunel, England.

[Burtnyk:1976]
N. Burtnyk, M. Wein, "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key-Frame Animation", CACM, Vol 19, #10, (Oct. 1976).

[Cardman:1975]
S. Cardman, "Time Management in a Real-Time Animation/ Graphics Environment", Computer Graphics, Vol 9, #1,

(SIGGRAPH 1975).

[Computer Image:1973]
Computer Image Corporation, "Coyote and Skunk", 7 min Colour, sound film, 1973.

[Cornwell:1971]
B. Cornwell, K. Cornwell, "Derivatives", a series of short colour, sound films, 1971.

[Cornwell:1976]
B. Cornwell, K. Cornwell, "Computer Animated Film: A Rat's Nest of Tradeoffs", NCC 1976.

[Csuri:1973]
C. Csuri, "Real-Time Film Animation", Annual report to the (U.S) National Science Foundation, Grant No.GJ-204, Jan. 1973.

[Davis:1968]
R.J. Davis, R. Nigel, W. Gruber, "A Model-making and Display Technique for 3-D Pictures", UAIDE 1968.

[Dwyer:1967]
W. Dwyer, "Windows, Sheilds and Shading", UAIDE 1967.

[DeFanti:1973]
T.A. DeFanti, "The Graphics Symbiosis System - An Interactive Mini-Computer Amination Graphics Language Designed for Habitability and Extensibility", Thesis, PhD., Ohio State University, 1973. (The main portion of the thesis appears in [CSURI:1973], pg. 5-116)

[DeFanti:1976a]
T. DeFanti, D. Sandin, R. Ainsworth, "Control Structures for Performance Graphics", Proc. ACM Symposium on Graphics Languages, April 1976, in Computer Graphics, Vol 10, #1, (Spring 1976).

[DeFanti:1976b]
T. DeFanti, "The Digital Component of the Circle

Graphics Habitat", NCC 1976.

[DeFanti:1977]
T. DeFanti, "Graphics Programming as a Performing Art", unpublished paper, 1977.

[Dowling:1976]
W.C. Dowling, "Computer Graphics and Design", Encyclopedia of Computer Science and Technology, Vol 5, eds. J. Belzer, A. Holzman, A. Kent, Marcel Dekker Inc., New York, 1976, pg. 447-472.

[Franke:1971]
H.W. Franke, Computer Graphics and Art, Phaidon Press Ltd., London, 1971; Translated by G.M. Metzger from, Computergraphik-Computerkunst, Verlag F. Bruckmann KG, Munich, 1971.

[Foldes:1974]
P. Foldes, "La Faim" ("Hunger"), 10 min colour and sound film, The National Film Board of Canada, 1974.

[Futrelle:1974]
R.P. Futrelle, "GALATEA: Interactive Graphics for the Analysis of Moving Images", IFIP 1974.

[Godfrey:1974]
Bob Godfrey and Anna Jackson, The Do-it-yourself Animation Book, BBC Publications, London, 1974.

[Goldberg:1976]
A. Goldberg, A. Kay (Eds), "SMALLTALK-72 Instruction Manual", Xerox Palo Alto Research Center, Technical Report SSL 76-6.

[Guerin:1973]
Marjorie Guerin, "A System for Computer Animated Film Production in a Batch Processing Environment", Thesis, M.Sc., University of Toronto, 1973.

[Halas:1970]
J. Halas, R. Manvell, Art in Movement - New Directions in Animation, Hasting House, New York, 1970.

[Halas:1971]
J. Halas, R. Manvell, The Technique of Film Animation, 3rd Edition, Focal Press, London, 1971.

[Halas:1974]
J. Halas (editor), Computer Animation, Hastings House, New York, 1974.

[Harlow:1965]
F. Harlow, J. Shannon, J. Welch, "Liquid Waves by Computer", Science #3688, (Sept. 3, 1965).

[Hershey:1967]
A.V. Hershey, "Calligraphy by Computers", U.S. Naval Weapons Laboratory Report No. 2101, Aug 1967.

[Hohl:1972?]
F. Hohl, "Dynamics of Disk Galaxies", 6 min colour, sound film, (1972?).

[Holman:1975]
D. Holman, B. Holman, "A Better Moustrap: CAESAR", Film-makers Newsletter, Vol 8, (Feb 1975).

[Holman:1975]
L.B. Holman, Puppet Animation in the Cinema - History and Technique, A.S. Barnes and Co Inc, Cranbury, N.J., and The Tantivy Press, London, 1975.

[Honey:1968]
F.J. Honey, "Computer Animation - A New Look", UNAIDE 1968.

[Honey:1971]
F.J. Honey, "Artist-Oriented Computer Animation", Journal of the Society of Motion Picture and Television

Engineers, Vol 80, (March 1971).

[Hopgood:1974]
F.R.A. Hopgood, "Computer Animation Used as a Tool in Teaching Computer Science", IFIP 1974.

[Kinsey:1970]
A. Kinsey, Animated Film Making, Studio Vista, London, 1970.

[Kitching:1973]
A. Kitching, "Computer Animation - Some New ANTICS", British Kinematography Sound and Television Journal, Dec 1973.

[Kitching:1976]
A. Kitching, C. Emmett, "The ANTICS Computer Amination System", Computer Graphics, Vol 10, #4, (Winter 1976).

[Knowlton:1964]
K.C. Knowlton, "A Computer Technique for Producing Animated Movies", SJCC 1964.

[Knowlton:1966]
K. Knowlton, "L6 - Bell Labs Low Level Linked-list Language", 16 min, B&W, sound film, 1966.

[Knowlton:1969]
K.C. Knowlton, L.L. Cherry, "FORTRAN IV BEFLIX", UAIDE 1969.

[Knowlton:1974]
K. Knowlton, W. Huggins, "Programming Languages for Computer Animation", in [HALAS:1974], pg. 45-54.

[Levine;1975]
S. Levine, "Computer Animation at Lawrence Livermore Labs", Computer Graphics, Vol. 9, #1, (SIGGRAPH 1975).

[NFB:1970]
National Film Board of Canada, "A Computer Controlled

and Operated Animation Stand", National Film Board of Canada, Technical Bulletin #6, 1970.

[Lipton:1972]
L. Lipton, Independent Filmmaking, Straight Arrow Books, San Francisco, 1972.

[LRG:1976]
Learning Research Group, "Personal Dynamic Media", Xerox Palo Alto Research Center, Report SSL 76-1, 1976.

[Mezei:1971a]
L. Mezei, "Art from Computers", 8 min colour, sound film, 1971.

[Mezei:1971b]
L. Mezei, A. Zivian, "ARTA - An Interactive Animation System", IFIP, 1971.

[Mezei:1973]
L. Mezei, T. Britton, "Artists to Computers", Proceedings of the 3rd Man-Computer Communications Seminar, National Research Council of Canada, May 1973.

[Negroponte:1976]
N. Negroponte, P. Pangars, "Experiments with Computer Animation", Computer Graphics, Vol 10, #2, (SIGGRAPH 1976).

[Newman:1973]
W.M. Newman, R.F. Sproull, Principles of Interactive Computer Graphics", McGraw-Hill Inc., New York, 1973.

[Noll:1968a]
A.M. Noll, "4-D Hypermovie", 5 min B&W, silent film, 1968.

[Noll:1968b]
A.M. Noll, "Computer Animation and the Fourth Dimension", FJCC 1968.

[Olshevsky:1970]
G. Olshevsky, "An Automated Aid to Visualizing Complex Hypersolids", Thesis, M.Sc., University of Toronto, 1970.

[Parr:1970]
C. Parr, "Reaction Dynamics", colour film, University of Toronto, 1970.

[Potel:1977]
M. Potel, "Real-Time Playback in Animation Systems", Computer Graphics, Vol 11, #2, (SIGGRAPH 1977).

[Pye:1977]
M. Pye, "STAR WARS - Magic in the Movies", Reprinted from the London Sunday Times in the Christchurch Press, July 26, 1977, pg. 17.

[Sandin:76]
D.J. Sandin, "The Analog Component of the Circle Graphics Habitat", NCC 1976.

[Schwartz:1974]
J.L. Schwartz, E.F. Taylor, "Computer Displays in Teaching Physics", in [Halas:1974], pg. 123-128.

[Sinden:1965]
F. Sinden, "Force, Mass and Motion", 10 min B&W, sound film, 1965.

[Speer:1974]
R.S. Speer, "Sources of Computer Films in the US, Canada, and Europe", Computer Graphics, Vol 9, #1, (SIGGRAPH 1975).

[Stephenson:1973]
R. Stephenson, The Animated Film, A.S. Barnes and Co Inc, Cranbury, NJ, and The Tavity Press London, 1973.

[Sutherland:1963]

I.E. Sutherland, "SKETCHPAD: a Man-Computer Communication System", SJCC 1963.

[Tilson:1975]
Michael Tilson, "Editing Computer Animated Film", Thesis, M.Sc., University of Toronto, 1975.

[Vanderbeek:1967]
S. Vanderbeek, K. Knowlton, "Man and His World", 1 min colour and sound film, 1967.

[Wein:1976]
M. Wein, N. Burtnyk, "Computer Animation", Encyclopedia of Computer Science and Technology, Volume 5, eds. J. Beltzer, A. Holzman, A. Kent, Marcel Dekker Inc., New York, 1976, pg. 397-436.

[Wessler:1973]
B.D. Wessler, "Computer-Assisted Visual Communication", Thesis, PhD., University of Utah, 1973.

[Williams:1971] R. Williams, "A Survey of Data Structures for Computer Graphics Systems", Computing Surveys, Vol 3, #1 (March 1971).

[Winkless:1968]
N. Winkless, P. Honore, "What Good is a Baby?", FJCC 1968.

[Youngblood:1970]
G. Youngblood, Expanded Cinema, E.P. Dutton and Co Inc, New York, 1970.

[Zajac:1964]
E.E. Zajac, "Computer-made Perspective Movies as a Scientific and Communication Tool", CACM, Vol 7, #3, (March 1964).

Appendix A

YACAS User Guide

Table of Contents

Section 1
Introduction

YACAS is a batch-oriented computer animation system. The following document is intended to serve as a guide to the use of the system. It is intended to be fairly brief; not to be a tutorial type introduction to the system.

The guide is organized into five sections and four appendices. Sections 1 and 2 are introductory, section 1 giving an overview of the system and section 2 explaining a number of the basic concepts upon which YACAS is based. Section 3 contains descriptions of each of the routines provided by YACAS, indicating parameters required, results returned, and the effects of each routine. Section 4 describes PLABAK, the film display portion of the system, while section 5 explains how YACAS can be extended by the user.

While the sections attempt to describe YACAS as a software system independent of the particular computer system supporting it, the appendicies give the necessary details (file names, magnetic tape numbers, work-flow language statements, etc) required to make active use of the system. As these details are prone to fairly rapid change, the user is cautioned to ensure that the information being used is current.

1.1 YACAS Overview

"YACAS" is a name applied to a set of Programs, Subroutines, concepts, and actions, that comprises a method for the production of animated films, using a computer (actually two computers). The computer's role in the process is to both create and manipulate the graphic images

required in the sequences specified, and to display them for viewing and photographing.

YACAS is a two-step process. The first step is to write a computer program; this program will create a data file (called a film file) in computer-useable form that contains information describing the film on a frame-by-frame basis. The second step is to use a program to view the film using the information in the film file. Usually the film won't be quite right the first time the program is run; the film is corrected by modifying the original program and repeating steps one and two until it is right. The "film" is copied from the film file to motion-picture film by repeating step two with a camera photographing each frame of the film from the computer display.

The next two sections describe these two steps in more detail.

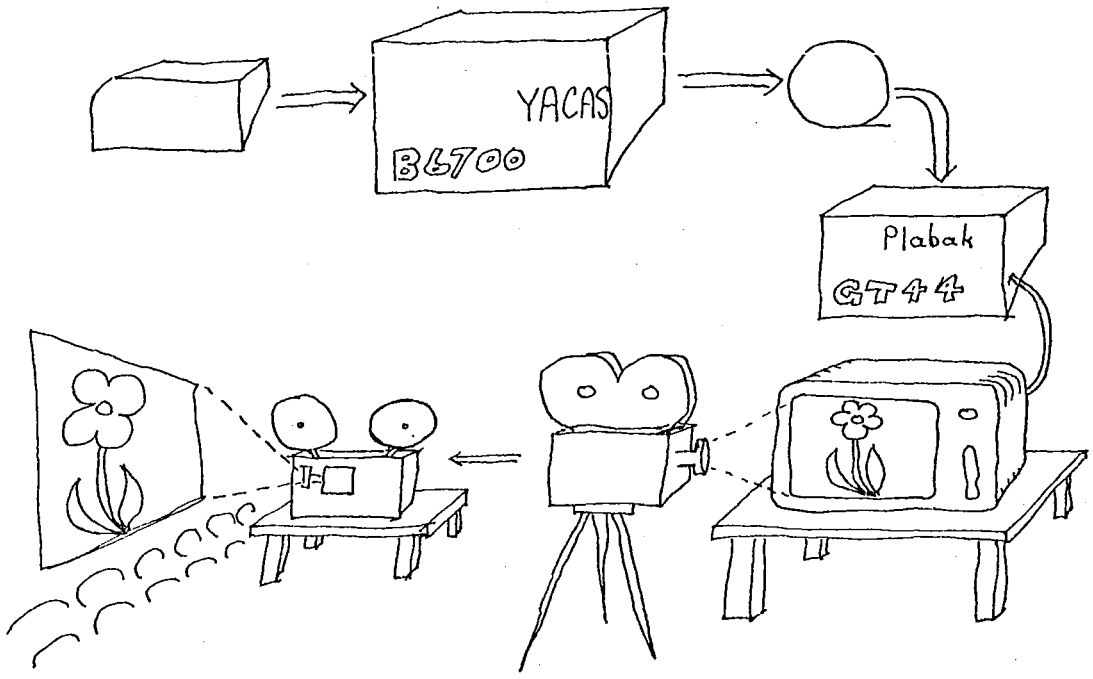


Fig. A.1-1
Filmmaking with YACAS

1.2 Step 1: Creating a Film File

The film file creation part of YACAS consists of a set of concepts about how pictures can be described and animated films made, and a set of subroutines that form a sort of "language" for describing the animated film to the computer in terms of these concepts. This "language" is used by writing a program (in Burroughs Extended ALGOL), calling the individual subroutines in the appropriate order to describe the film. Alternatively, a 'program' may be written using just the YACAS commands with appropriate parameters, by using the FILMMAKER program to interpret this 'program' to the computer.

Whichever approach is used, the YACAS system does the following things to help the user make the film desired.

(1) Picture information is stored in data structures created and maintained by YACAS. These pictures may be simple, articulated, and/or shared; Section 2.1 describes the picture concepts of YACAS. Pictures may be created or deleted at any point in a program.

(2) Pictures may be manipulated with either distorting or non-distorting transformations provided by YACAS or by the user (see section 5). These transformations may be applied 'statically' (between film frames), or 'dynamically' (over a period of time).

(3) Film sequences can be described which involve many simultaneous actions on an arbitrary number of different pictures. These actions may start and stop at different times, proceed at different rates, and may be wholly or partially independent of each other.

(4) Certain actions can be carried out superimposed upon all the other actions being performed in a sequence; these are referred to as 'camera' actions. Those implemented are the ZOOM, PAN, FADE and DISSOLVE commands.

(5) Two output media are available for selection: display (from which a film may be photographed) or paper-plotter.

The latter is normally used in "Cartooning" mode to draw selected frames from a film for debugging and/or documentation purposes. Either or both media may be selected for a given run.

When the display is selected for output, YACAS produces a 'film file'. This file contains the information required to construct a display of the film. The display of film file images is the task of step 2.

To provide YACAS routines with pictures to manipulate, two methods of picture description are provided. The first method involves the use of a set of commands to describe a picture on a vector-by-vector basis as part of the program. A picture may have an arbitrary number of vectors; vectors may be visible or invisible. The second method involves the creation of a data file containing vector-by-vector picture descriptions to be read into the program by the "READCEL" command.

Admittedly these facilities are just barely adequate. Enhancements in this area would receive high priority in future versions of the system.

1.3 Step 2: Displaying the Film File

The film file production step of the YACAS system is designed for batch computing. With appropriate hardware (a Computer Output Microfilm unit with graphics capability and sprocketed 16mm film output), the whole operation could be carried out as a batched process. However, computer animation involves repeated attempts at film production with each new step refining the film until it "looks right" (the prime criterion of correctness). Since the computer-to-film stage of the process is often slow, relatively costly, and highly dependent on specialized equipment, it is desirable to be able to preview the film before committing it to film. This cannot be done effectively in a batch environment.

YACAS is designed around the availability of both batch and interactive computing facilities; preferably both in the same organization. The interactive facility should have a refreshed graphic display, and should be (as a preference) a single-user stand-alone computer system with modest main memory (56k characters) and disk storage (5.4M characters).

The function of the display step is to provide the film-maker with the ability to view the contents of a film file. This file may be repeatedly viewed, at different speeds; portions of the file may be selected for particular attention. The film may be stopped at any point for detailed inspection of its individual frames.

In the particular version of YACAS described by this manual (that at the University of Canterbury, Christchurch, New Zealand), the computer used for the display portion is a Digital Equipment Corp. GT44 (PDP-11 with display), while that used for the batch portion is a Burroughs B6700.

Section 2
YACAS Concepts

Although YACAS can be described in terms of the functions of each of its subroutines (see Section 3), a proper understanding depends on familiarity with of several underlying concepts. This section describes those concepts.

2.1 Pictures and Cels

When one takes a photograph of something, a flower for example, one obtains (after appropriate processing) a "picture" of the flower. One can obtain several copies of the picture by printing the negative several times. Each copy may be the same as the others, or may differ from them in certain ways: a specific copy may be small, off-center, or twisted relative to the edges of the printing paper (among other possible differences). Each copy is an instance of the picture; each looks the same as the others, though may differ in position, size or orientation.

YACAS makes a similar conceptual distinction between the appearance (shape) of an image and its position-size-orientation. In YACAS an entity called a "cel" is roughly equivalent to the negative of the example above. An instance of a cel is called a "picture". There are a number of similarities between the YACAS concepts of cel and picture, and the negative and print of the photography example; these similarities will be drawn upon to help make the YACAS ideas clear.

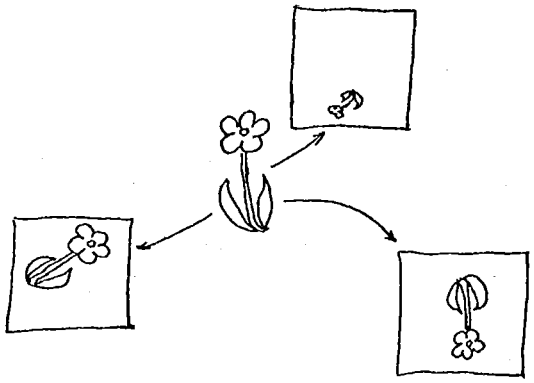


Fig. A.2-1
Cel with 3 pictures

2.1.1 Cels

As with the negative, a cel contains the basic information about the shape of an object. This information is independent of any specific picture; all pictures made from the same cel will look alike. An arbitrary number of copies may be made from a single cel (and unlike the negative, there is no worry about dust, finger prints, or scratches).

If a negative is to be copied onto a large sheet of photographic paper, there are four principle things that can be varied to make one picture a little different from another: the size may be altered (assuming an enlarger is available), the image may be positioned anywhere relative to the edges of the paper, it may be twisted relative to the edges, and its intensity may be altered (one may be darker or lighter than another). These four things are available in YACAS to make one picture (instance of a cel) different from another, except that in YACAS, the equivalent of the paper is the "drawing space" (see section 2.2.4).

2.1.2 Pictures

Pictures are instances of cels. There may be only one picture associated with a single cel, or there may be several. YACAS provides for an arbitrary number of pictures (currently 100) and an arbitrary number of cels (limited by memory resources). Each single picture is an instance of a single cel. Returning to the photographic example, if one supposes having an arbitrarily large supply of paper and an arbitrarily large supply of negatives (and plenty of time for the processing), and make the further stipulation that each piece of paper will be allowed only one negative printed on it, then the comparison between pictures and cels, and negatives and prints holds: each negative (cel) may be used to make an arbitrary number of prints (pictures) each of which may differ from the others in detail such a position size, etc.; each print (picture) is produced from a single negative (cel).

Information regarding the position, size, orientation, and intensity of a picture goes relates directly to the picture and goes into a table of picture information. Information regarding the shape of the image goes into a cel table.

2.1.3 Articulated Pictures

If we remove the stipulation that each piece of paper be allowed only one negative printed on it, then we get close to the YACAS concept of an "articulated" picture.

In YACAS, an articulated picture is one in which two or more pictures are "joined" together and may be treated as a single picture. In YACAS this is done on a hierarchical basis: in the simplest situation (two pictures), one is designated the "root", with the other attached to it. When there are more than two pictures, there is still one root, with the others attached to it (and not to each other except

as noted below). A common terminology used to express these relationships is to call the root a "father" picture, while those attached to it are called "sons". A father may have an arbitrary number of sons.

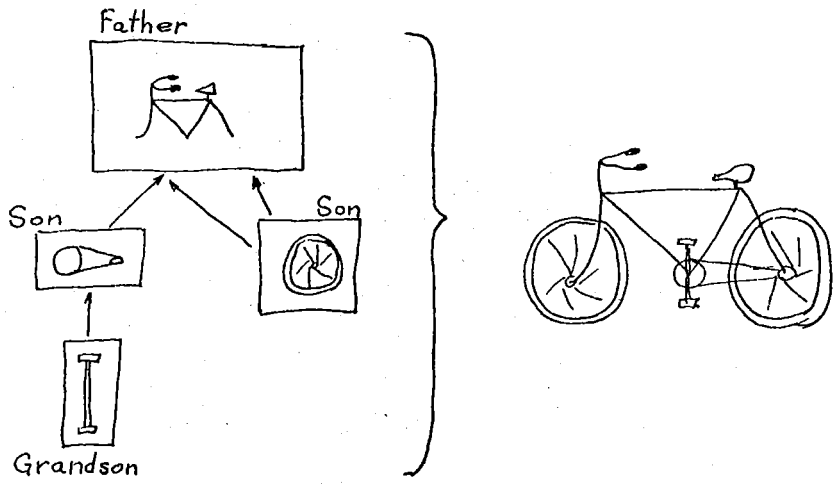


Fig. A.2-2
An articulated picture - bicycle with separate sub-pictures.

Returning again to the photographic example, the group of pictures referred to as an "articulated" picture would be comparable to printing several negatives onto a single sheet of paper. The resulting composite picture can be treated as one (being a single sheet of paper) yet is composed of several separate images. The idea of the hierarchy is a little more difficult to include in the example. If the piece of paper had one negative printed first, then subsequent negatives were printed relative to the first one only (without regard for any others there), then the relationship of these others to the first one is similar to that between the father picture and its sons.

YACAS articulated pictures are not restricted to a single "level" of connectivity. Any "son" picture may itself be a "father" to an arbitrary number of other

pictures; its sons may themselves have sons. This may continue to an arbitrary depth.

Although the method of connection between pictures in an articulated picture will be discussed in more detail in the next section (2.2 - Picture Geometry), something of the nature of the connection can be described here. An articulated picture can be treated as a single picture. It may be moved, scaled, rotated etc. by moving, scaling (or whatever) the root picture. If the position of the root picture is changed, the positions of all the other pictures are changed; all the other pictures in the group will bear the same positional relationship to the root as before, but will differ in absolute position. Similarly, changing the size or orientation of the root will change the size or orientation of the sons, but the relative size or orientation will be preserved. A son though, has a certain degree of freedom within the structure: although its position is determined by its connection to its father, its size and orientation relative to its father may be altered. There is only one restriction: each picture in the set except the root, may and must have only one father picture; the root may not have a father.

2.1.4 Summary

To summarize, there are two basic parts to YACAS images - pictures and cels. Cels contain the information about the shape of an object; a picture is an instance of a cel, defining its position, size, orientation, and intensity. There may be an arbitrary number of instances of a single cel, each being a single picture; a (simple) picture is an instance of a single cel.

Pictures may be grouped into a hierarchical structure referred to as an articulated picture. In an articulated picture, each picture except the root must have one and only one father; each (including the root) may have an arbitrary

number of sons. The position, size orientation, and intensity of an articulated picture may be controlled by controlling those parameters of the root picture. Changes to the size, orientation, and intensity of a son may be made; these affect the size etc. of the son relative to the son's father.

2.2 Picture Geometry

In section 2.1, the idea that cels contain the information about the shape of a picture was introduced. This section will describe the method used, and relate this to the images produced on a physical medium (display, film or paper).

2.2.1 Points, Vectors and Curves

The definition of YACAS pictures is based on two-dimensional Euclidean geometry. A "point" is a pair of numbers representing the coordinates of a location within a "drawing space" (see 2.2.4). A "vector" is a straight line joining two points. A vector is "visible" when it would appear if drawn on an output device; it is "invisible" when it would not be visible (when the "pen" used to draw the vectors is moved to the start of a visible vector). A "curve" is a set of vectors, beginning with an invisible vector from the end of the previous curve, followed by an arbitrary number of connected, visible vectors. A cel may be described with an arbitrary number of curves.

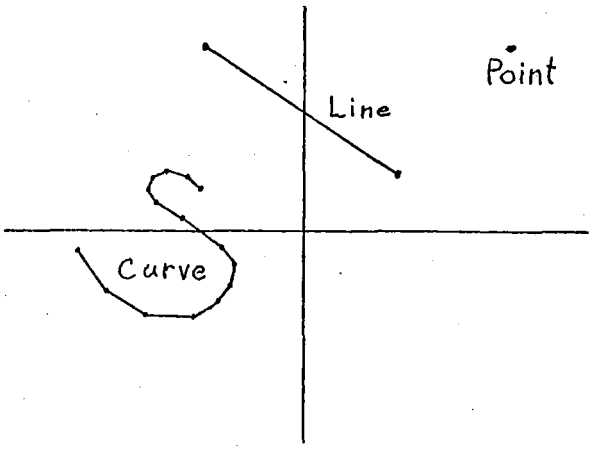


Fig A.2-3
Points, Lines and Curves

2.2.2 Defining Pictures and Cels

Pictures are described to the YACAS system by first opening them, describing them, then closing them. The principal routine involved in describing a picture is the VECT routine which defines the position of a point in the picture description, and indicates whether it is connected to the previous point in the picture, or not (thus indicating whether it is the end-point of a visible or invisible vector).

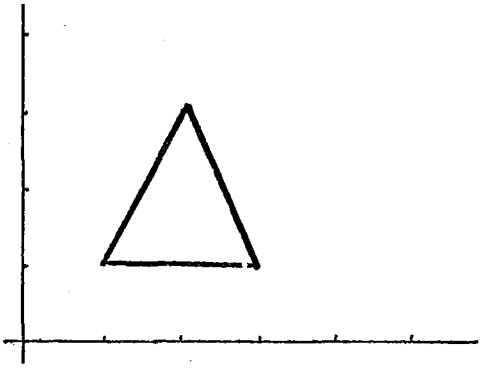
The OPENP routine informs the YACAS system that a new picture is about to be described. OPENP returns the picture's identifying number; this number should be used to identify the picture to the other YACAS routines. Following the OPENP, an arbitrary number of calls may be made to the VECT routine. These define all the points in the picture, and the way they are connected together. When the picture has been fully described, it must be closed using the routine CLOSEP.

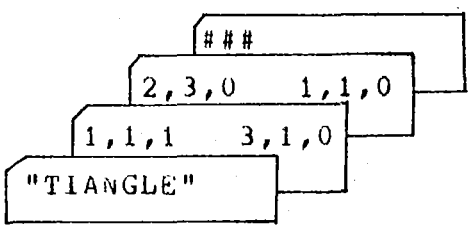
The shape information from the set of points is stored as the cel description. The position of the cel is

extracted from these points and is stored as part of the "picture" description. Initially the picture description is given the values (1.,1.) for the X and Y size factors, 0. for its orientation, and an intensity of 6. The effect of these values is that the picture, if drawn, would appear at the position, with the size and orientation inherent in the vector-by-vector description.

The cel information may be specified in two other ways. The routine USECEL indicates to YACAS that the picture just opened is to share the use of the cel with another picture (previously defined). The parameters of the USECEL routine indicate which cel is to be shared (by specifying the number of a picture that already uses it), and allow the user to specify the position, scale factors, and orientation of the new picture directly. An alternative method of defining a cel is available through the READCEL routine. The cel descripion may be prepared as a set of data cards. These are read and used to define a cel via the READCEL routine. The data cards may be prepared in a free format; they consist primarily of triples of numbers for each point - the triple provides the parameters for the VECT routine. A character string may be associated with the cel, primarily for documentation purposes, and is returned in the array passed as READCEL's parameter. The rules for forming a cel description for READCEL may be found in Appendix 4.

```
PIC := OPENP;
  VECT(1,1,1);
  VECT(0,3,1);
  VECT(0,2,3);
  VECT(0,1,1);
  CLOSEP;
```






```
PIC := OPENP;
    READCEL(NAME);
    CLOSEP;
```

Fig. A.2-4
Picture creation using VECT and READCEL

To define an articulated picture, a similar OPENP, describe cel, CLOSEP approach is used. The difference lies in the fact that, to describe a picture as the son of another picture, the father is opened, its cel described, then the son is opened, described, and closed, then finally the father is closed. When there are to be several sons, the father picture is opened and described, then the sons are opened, described and closed one after the other, before finally closing the father. An alternative method of defining a picture as the son of another picture is to define it earlier, then use the routine INCLUDEPIC before closing the father. A picture to be included in this way must be either a simple picture or the root of an articulated picture.

```
FATHER := OPENP;
  VECT(1,1,1);
  VECT(0,3,1);
  VECT(0,2,3);
  VECT(0,1,1);
 SON := OPENP;
   VECT(1,2,0);
   VECT(0,4,0);
   VECT(0,4,2);
   VECT(0,2,2);
   VECT(0,2,0);
   CLOSEP;
 CLOSEP;
```

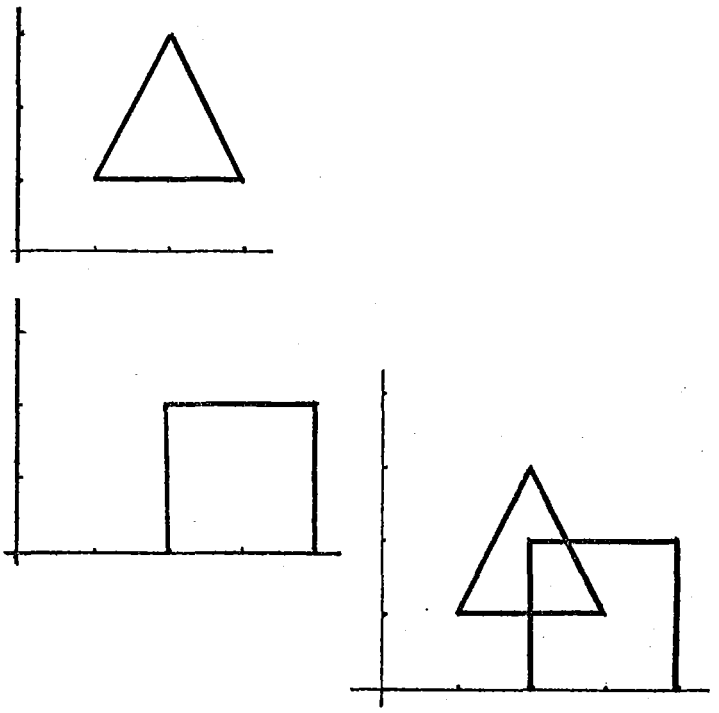


Fig. A.2-5
Articulated picture creation.

2.2.3 Pivot and Attachment Points

When a two-dimensional picture is rotated, it does so around some point. In YACAS, this is called its "pivot point". Each cel has its own pivot point; all non-distorting actions on the cel (eg: MOVETO, SCALEBY, ROTATETO) are performed around this point. The pivot point need not exist in the cel description as a specific point. It merely serves as a reference point for rotations etc.

The location of the pivot point can be determined by YACAS from the original picture description, or it may be specified by the user. When YACAS determines the pivot point, it uses the "geometric center" of the described picture. This point is the mid-point between the extreme co-ordinate positions in the X- and Y- directions. When the pivot point is specified by the user, it need not lie at the picture's geometric center; it need not lie within the bounds of the picture at all.

When dealing with articulated pictures, some means is needed to connect the component pictures together. In YACAS this is done using the pivot points and "attachment" points. An attachment point is just that: a point in one picture (the father) to which another picture (a son) can be attached. In articulated pictures, the pivot point may be thought of as an attachment point with some special properties; sons may be attached to their father's pivot point, or to an attachment point. The attachment is accomplished by positioning the son so that its pivot point is located at the same location as the attachment point to which it is attached.

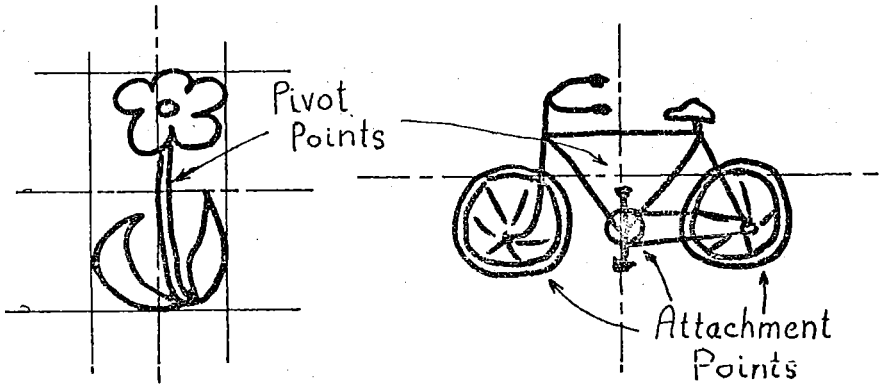


Fig. A.2-6
Pivot and Attachment points

Attachment points are defined when a picture's cel is defined as part of the cel description of the picture. They are defined as the last set of points in the cel description, and are distinguished from the detail (and outline) points by preceding them with a call to BGNATT. They are specified using the routine VECT; in this use, the "type" parameter of VECT is ignored. When a "son" picture in an articulated picture is defined, the point to which it is attached is indicated through a call to ATTACHTO. This indicates the father's attachment point by number: the attachment points are numbered from 1 in the sequence defined; the pivot point is number 0.

In special circumstances a cel is allowed to have no visible vectors. The situation where this may be meaningful arises when the cel contains only attachment points. Here, the cel serves to join two or more pictures together, yet is itself not visible.

2.2.4 Drawing Space, Windows, and Viewports

All that has been mentioned above about points, vectors, etc. has only referred to them as having some position. The positions are represented as pairs of numbers, themselves representing "distances" along two orthogonal axes (called the X and Y axes) from the point of intersection of the axes. In "normal" geometry this is as specific as things need to get: the numbers used to represent points are not limited in value. In YACAS, because it is implemented on a computer, there is a limit to the values that may be used to represent points. In YACAS, the numbers used to represent points are ALGOL REAL numbers; the values are limited to the range (approximately) +/-10**(+/-38).

If pictures are thought to be drawn on a flat surface, in "normal" geometry, the surface would be infinite in extent; in YACAS the surface is finite. This finite surface is referred to as the "drawing space".

Although the drawing space is finite in size, it is nevertheless quite large. Normally we are interested in only those pictures positioned in a small area of the drawing space. This area of interest could be called the field-of-view, but is more commonly called the "window". When a picture is to be drawn on an output device, only that portion of it that lies within the window will be drawn; the rest will still exist - just not be drawn. In YACAS the default window is a rectangular area with its lower left corner at (0,0), lower right at (20,0), upper right at (20,15), and upper left corner at (0,15). Normally the

window is specified by only its lower left, and upper right corners, thus the default window is defined by the points (0,0) and (20,15). Commands exist to change the position and size of the window (but not its orientation).

When drawing pictures that lie within the window on some output device, some portion of the drawing area of the device is used. This portion is called the "viewport". The viewport is a rectangular area whose sides have lengths proportional to the lengths of the sides of the window.

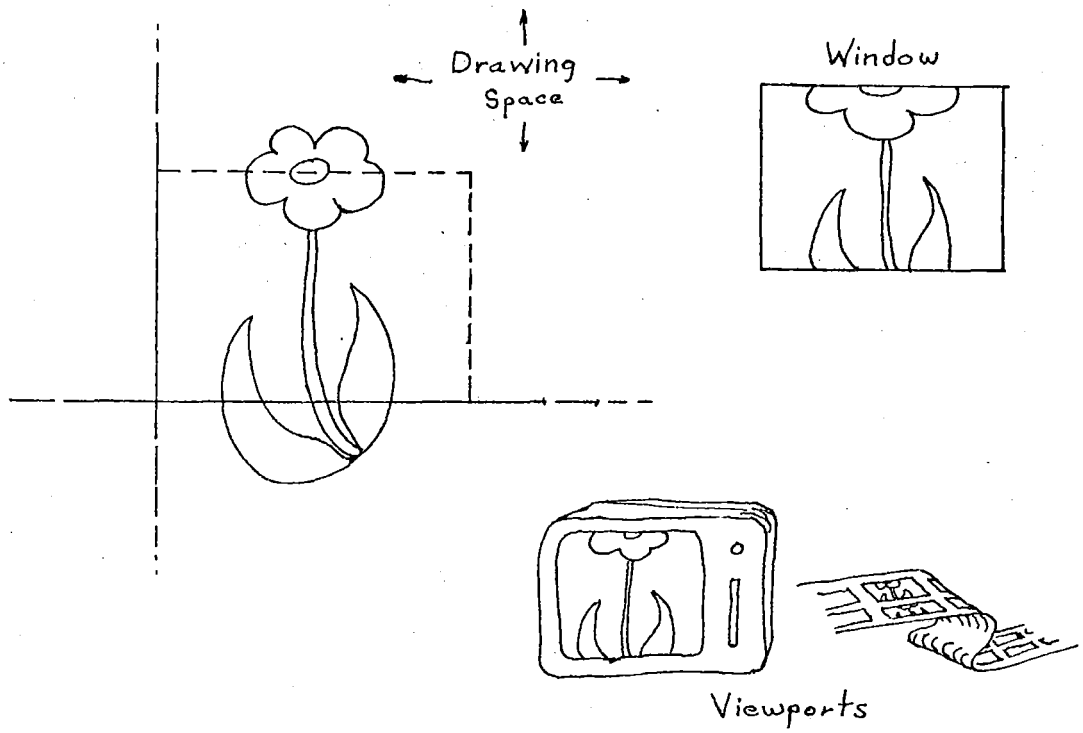


Fig. A.2-7
Drawing space, Windows, and Viewports

In animated film making, the primary unit of interest is the "frame" - the image of a set of pictures (in our case, those within the window) at some instant in time. A film is made by drawing a succession of these frames. When drawing to the display, individual frames are drawn in the same place - in the viewport - but are separated from each other by time. When drawing to the plotter, time is not a

useful way to separate the frames but position is. For the paper plotter, the viewport's size and shape are generally fixed (though the size may be changed), but its position changes in a regular manner. Frames (viewports) may be drawn in groups of one, two, three, or four across the width of the plotter, with the groups repeated along the length. The size of the viewport is determined by the number to be drawn in a group. This is specified through the CARTOONING command.

2.3 Time Units and Intervals

In making an animated film, time is one of the most crucial of the elements to be controlled by the animator. Time is controlled in the sense that the images are changed in a controlled manner over time. The following subsections discuss the measurement of time, and the specification of timed sequences (intervals) as they are implemented in YACAS.

2.3.1 Time Units

Any film, whether live-action or animated, consists of a sequence of images. The relationship between the time represented by a sequence of pictures and "real" time is determined by the rate at which the film is projected. Film recorded on 16 or 35mm film is usually projected at 24 frames per second (though occasionally 16mm film may be projected at the "silent" rate of 16 frames/sec). The increasingly popular 8mm films are usually projected at 18 frames/sec. An underlying assumption of YACAS is that the film produced by it will be projected at the 24 frames/sec rate. However, since YACAS only produces sequences of frames, the user may project them at whatever rate seems suitable.

Within YACAS, time is measured in discrete units called "frames". Each frame (unit of time) corresponds to one frame of film (image on film). The user may wish to work with time units that correspond to groups of frames; the UNITS command provides the mechanism for this. The parameter to the UNITS command indicates the number of frames there are to be in the time units he wishes to work with. Thus "UNITS(2.);" indicates that the user wishes to work with time units that are 2 frames long. YACAS allows two symbolic literals as the parameter for the UNITS command: "FRAMES" (representing the value 1.), and "SECS" (representing the value 24.).

The time units defined with the UNITS command are used with three other commands. CARTOONING uses the time units to set a selection criteria for producing paper plots (see its entry in section 3.1). INTERVAL, and ROLLIT will be discussed below (sections 2.3.2 and 2.3.3).

One other aspect of film time deserves mention. YACAS is usually instructed to carry out some action on a picture over some interval - some number of frames. YACAS may, for example, have been told to move a picture from one position to another in, say, 10 frames. To carry this out, YACAS will compute the position of the picture ten times - once for each frame. Each of these is a "computed frame". Each computed frame may be displayed, or copied as a film frame an arbitrary number of times. The method of specifying this is with the FILMSPEED command. FILMSPEED indicates the number of copies to be made of each computed frame; the default is 1.

2.3.2 Intervals

An animated film involves sequences of images changing in some way. If the image is a single picture, that picture may change in shape, size, position, orientation, intensity, or some combination of these. The change takes place over

some finite length of time; this time is referred to as an "interval". An interval is clearly defined by a starting time and a duration.

An interval is simply a segment of time within the course of a film (a film could consist of a single interval, but usually encompasses several). Within a single interval one thing may happen to one picture (ie: a picture may move from one position to another); several things may happen to a single picture (a picture may move, and change size simultaneously); or several things may happen to several pictures (picture A may move while picture B changes size, while picture C changes shape and orientation). In general, an arbitrary number of things may happen to an arbitrary number of pictures within a single interval.

A film usually consists of several intervals. The intervals may occur sequentially - one after the other - or they may overlap in time to any extent. For example: one interval may start, then before it finishes a second may start. Before either of these end, a third may start, do its thing, then end; the first may then end, then finally the second. The order of intervals starting and stopping does not matter to YACAS, though it does to the animator's intent.

Where several changes are to be made to a single picture, these changes may be made all in the same interval, or may take place over several. The intervals (if more than one is involved) may overlap in time to any extent necessary.

The method of associating motions with intervals is as follows. The start of an interval is signalled through the use of the INTERVAL command. This specifies when the interval is to begin and how long it is to last. All YACAS commands subsequently issued, that cause changes to be made to pictures or the window, will be queued for later

execution (when the interval is to be carried out). An interval description is terminated by either the issuing of another INTERVAL command, the issuing of the SETSTATIC command, or the issuing of the ROLLIT command (see below).

Some YACAS commands produce their effect immediately. The OPENP command immediately opens a new picture, and the VECT command immediately adds a new point to the current cel description. Some YACAS commands produce their effect immediately in some situations, and defer their effect till later in others. The motion commands such as MOVETO, SCALEBY and ROTATETO behave in this way. Some other commands - INTERPOLATE and the dynamics functions - operate in the deferred manner only.

For the motion commands, there are two modes of operation: "static" and "dynamic". The motion commands operate in static mode when issued outside an interval description, and operate in dynamic mode when issued within an interval description. A motion command is operating in static mode when it produces its effect immediately; it is operating in dynamic mode when it defers is effect. When a motion command is issued in dynamic mode, an entry is made in a queue of motions to be performed. This queue is called the "agenda". Motions that are queued in the agenda are carried out by issuing the ROLLIT command.

2.3.3 ROLLIT and the Agenda

When YACAS is being given an interval description, it builds a list of the things it has deferred to do later. This list is called the agenda. The agenda will contain a description of all the things YACAS is to do over several intervals; perhaps over the whole film. But the agenda has a finite length (arbitrarily set at 50 entries) and may not have room to hold the interval descriptions of the whole film. To tell YACAS that a portion of the agenda is to be carried out, the ROLLIT command is used. When YACAS

receives this command, it sorts the agenda entries into starting-time order, then carries them out over the interval specified in the ROLLIT command. When finished, those agenda entries that have been completed are removed; any not completed (either not yet started or only partially completed) are left.

ROLLIT has as its parameters a starting time and a duration. These represent a section of the film to be produced. The starting time may be "now" (see below), some time before "now", or sometime after "now". Whenever YACAS executes a ROLLIT command and finds nothing in the agenda to do for a frame, it produces a blank computed frame. When there are things to do it does them then draws the pictures operated on in the frame. If a command in the agenda is to start before the interval specified in the ROLLIT command, it is carried out up to the time of the ROLLIT interval without any pictures being drawn.

"Now" is one of those elusive time concepts that lie at the core of YACAS. Time in YACAS progresses in leaps and bounds. Time (in the form of film produced) progresses only when a ROLLIT interval is being carried out; time "stands still" otherwise. "Now" lies at the boundary between time that has passed (film frames that have been produced), and time that has yet to come. All specified times (in INTERVAL, ROLLIT and CARTOONING) are relative to this boundary. Thus "INTERVAL(0, 2);" indicates an interval that is to begin "now" and last for 2 time units; "ROLLIT(1, 12);" represents a ROLLIT interval starting 1 time unit from "now" and lasting 12 time units. INTERVAL and ROLLIT allow the literal "*" to represent "now". ROLLIT allows this same literal ("*") as its duration to represent 'everything in the agenda'.

One final point. For a picture to appear in a frame, it must be drawn. When commands are issued within intervals and executed via ROLLIT, the pictures directly affected are

drawn automatically by YACAS. If a picture is to appear in a frame but has nothing to be done to it to get it drawn automatically, it may be drawn explicitly with the DRAW (or DRAWF) command. The DRAW command is like the MOVETO command in that it may be carried out in both static and dynamic modes. In dynamic mode the command (in both forms: DRAW and DRAWF) is placed in the agenda to be carried out for every frame of the current interval. In static mode, the DRAW form of the command indicates that the picture is to appear in the current frame; the DRAWF form does the same, and also indicates that the full frame (including any entries in the agenda) is to be actually drawn (time progresses by one frame).

2.4 Motion Dynamics

In an animated film, if a picture is moved over a specified interval, the rate of motion may be constant or may vary. If the rate is constant, the motion occurs at a steady speed; if the rate varies, the speed of motion varies. The rate at which a motion is applied is referred to as the motion's "dynamics".

In YACAS, motion dynamics are controlled by subroutines. Three are available: LINEAR, CUSHIONED, and SINUSOID; these routines are discussed below.

Any of the YACAS motion commands (MOVETO, SCALEBY, INTERPOLATE, PAN, DISSOLVE, etc) may be controlled by any of the dynamics routines. To indicate which dynamics are to be applied to a specific motion, follow the motion command with the desired dynamics command:

```
eg: INTERVAL (0.,12.);
    MOVETO (PIC,NEWX,NEWY); CUSHIONED;
```

If no dynamics command is specified, LINEAR dynamics will be used.

2.4.1 LINEAR

When a motion occurs with linear dynamics, the amount of change from one frame to the next is constant (fig. A.2-8). If, for example, a picture is to move ten units of distance in twenty frames with linear dynamics, then for each frame, it will move 1/2 unit.

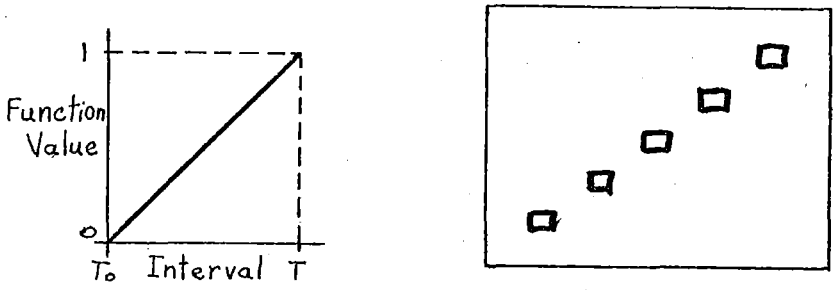


Fig. A.2-8
LINEAR dynamics: graph and example

Linear dynamics have a noticable side effect that is sometimes undesirable: the abruptness with which the picture assumes its speed of motion. This is particularly noticable when the picture is either at rest to begin with, comes to rest at the end of the motion, or both. The motion appears to jerk at the beginning, progress smoothly, then jerk at the end. This jerkiness increases when the speed of motion before the interval is much different from the speed during the interval; the jerkiness is reduced when the speed difference is small.

2.4.2 CUSHIONED

A more "natural" way for an object to acquire motion is for it to accelerate to a reasonably constant speed, then to decelerate when the motion is completed. So too for motions applied to pictures in an animated film.

When CUSHIONED dynamics are applied to a motion, the speed of motion starts small, then over a few frames gets faster until it is nearly constant, than a few frames before

the end the speed gets slower until the motion stops. For most motions this will give a fairly "natural" appearance to the motion, without the jerkiness often found with LINEAR dynamics (Fig. A.2-9).

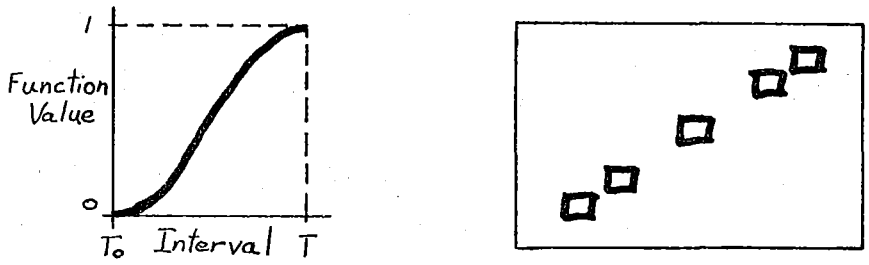


Fig. A.2-9
CUSHIONED dynamics: graph and example

2.4.3 SINUSOID

Cyclic motions occur in a number of situations: swinging pendulums, vibrating strings etc. YACAS provides a mechanism for specifying cyclic motions in the form of the SINUSOID dynamics function.

When SINUSOID dynamics are applied to a motion, the motion progresses from its starting state to its extreme and back to its start as one cycle (Fig. A.2-10). As an example, if the command sequence

```
MOVETO (PIC,X2,Y2);
SINUSOID (1);
```

was executed over some interval, the picture PIC would move from its initial position (say X1,Y1) to the position X2,Y2 then back to X1,Y1 in the course of the interval. The apparent moves from X1,Y1 to X2,Y2 then back to X1,Y1 would appear to have cushioned dynamics.

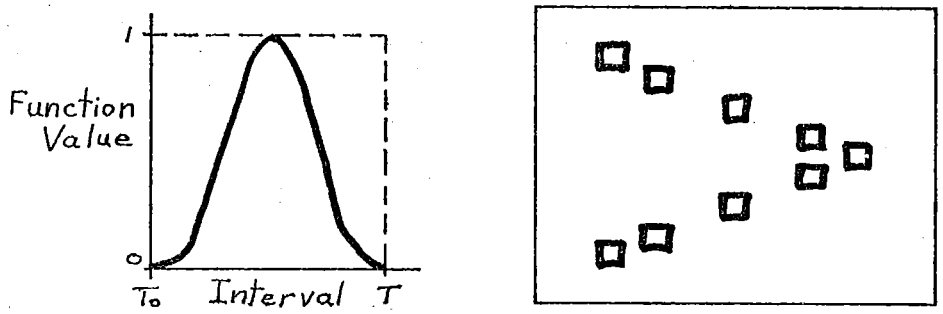


Fig. A.2-10
SINUSOID dynamics: graph and example
(a single cycle)

A property of cyclic motions is the length of time a single cycle takes. The parameter to the SINUSOID command indicates the numbers of cycles that are to occur in the interval. Thus SINUSOID(1) indicates that the motion is to go through one cycle during the interval; SINUSOID(2) indicates that there are to be 2 cycles in the interval. The parameter value may be a fraction; thus SINUSOID(1.5) indicates that the motion is to go through 1.5 cycles in the interval (in the MOVETO example above, the picture PIC would move from X1,Y1 to X2,Y2, back to X1,Y1, then finally back to X2,Y2).

2.5 Interpolation

One of the most common operations involved in the production of an animated film is the change of shape of one picture into another. In cel animation this is called "in-betweening". If the word "picture" is taken to mean a whole film image (frame) then frequently the change from one picture (frame) to the next may be produced by a combination of MOVETOs, SCALEBYs, or ROTATETOs of one or more of the component pictures. Frequently, though, such simple motions do not suffice and, at the individual component level, a picture must change its shape. The algorithm used to change the shape of a picture is called "linear interpolation". Fig A.2-11 shows an example of the use of the interpolation

algorithm (used to produce a single picture) in which the outline of a running man is changed into that of a bottle of coca-cola, which is then changed into an outline of the continent of Africa.

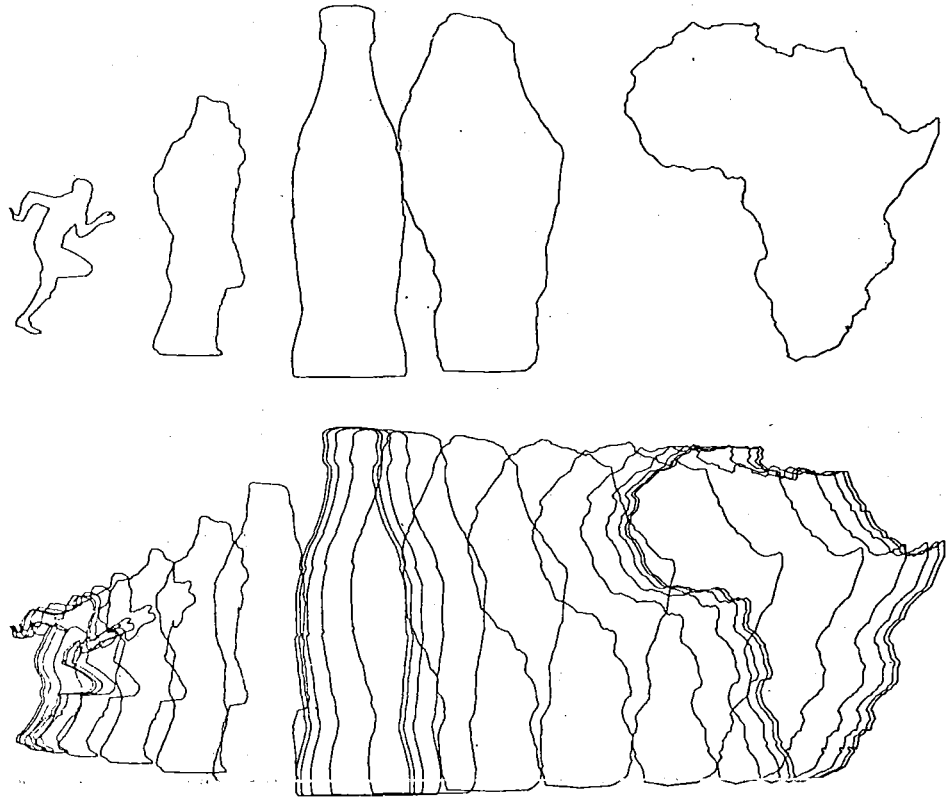


Fig. A.2-11
"Running Cola is Africa"
- Computer Techniques Group
(Reproduced from [Franke:1971], pg.34)

An example from classic fairy tales may serve to illustrate the intent. A beautiful princess chances upon an ugly toad and learns that the toad is, in fact, a handsome prince turned into a toad by a wicked witch. A kiss from the princess will reverse the spell; the princess kisses the toad. The toad becomes a prince. If this was being animated, the change from toad to prince could be handled in two ways: the toad could instantaneously change - accomplished by simply replacing the toad cel by one of a prince; alternatively, the change could be more gradual, taking from several frames to several seconds to transpire.

The between frames can be produced via interpolation.

The YACAS subroutine INTERPOLATE implements one form of the interpolation algorithm. This subroutine will change the shape of the cel associated with PIC1 to the shape of the cel associated with PIC2. The position, size, orientation and intensity of PIC1 remain unchanged (unless other subroutines are affecting these parameters). If the pictures indicated are articulated pictures, only the cels of the named pictures are affected; the cels of the sons are not.

The algorithm used imposes certain conditions on the nature of the cels involved, and may produce some unexpected side-effects.

The basic algorithm proceeds as follows: each point in the starting cel is matched with one in the final cel. For each intermediate frame in the sequence, a new cel is created by calculating the new position of each point from the original cel as it moves along a straight line from its original position to its final position (that of its corresponding point in the final cel).

The correspondence between points is made by associating the points of the starting picture's cel one-by-one with those of the final picture's. The cel descriptions of the two pictures must have exactly the same number of points. Fig A.2-12 shows the interpolation of a square into a star shape. The path of one of the points is shown as a dashed line.

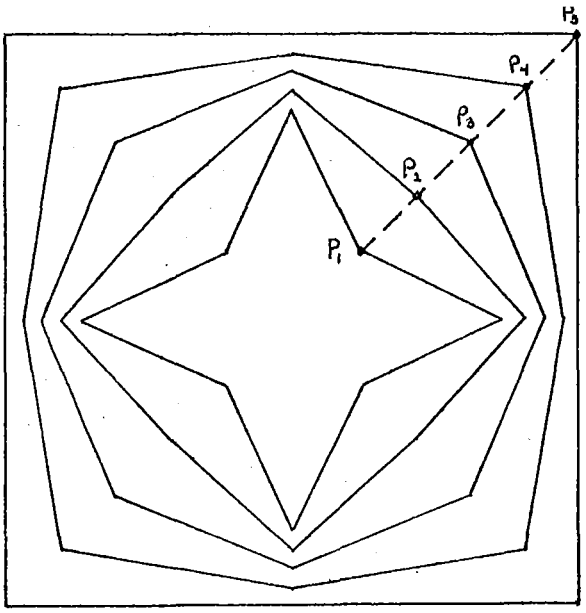


Fig. A.2-12
Linear Interpolation

The YACAS user is not directly responsible for ensuring that the two cels involved meet the "equal number of points" requirements. When INTERPOLATE is called, YACAS checks the two cels involved to see whether the numbers of points are the same. If not, extra points are added to the cel that has too few. The shape of the cel is retained by adding the points along the visible vectors of the cel; the added points are distributed evenly along the total length of the visible vectors. If the cel that is altered is the final cel, a dummy picture is created by YACAS and given the new extended cel; this dummy picture will be deleted at the end of the interval. If the starting cel is the one given the added points, then no dummy picture is created. At present, YACAS makes no distinction between detail and outline points when counting points in cels (subrequent versions may change this policy); where the number of attachment points differs, added points are located at the cel's pivot point (0,0).

A restriction imposed on use of the INTERPOLATE command is that an individual cel may be interpolated into a new shape only once in any particular interval. The user could inadvertently violate this rule if two pictures used the same cel, and one of these pictures was used in one INTERPOLATE command, and the other picture in another. The COPYPIC command is available to allow cels to be duplicated and can be used to get around this restriction.

Three consequences must be kept in mind.
(1) Since the picture's cel is being changed, any other pictures using that same cel will be changed also (in the example suggested above, if there were several toads represented by one cel, all would change to princes when the princess kissed one of them).
(2) YACAS currently makes no effort to ensure that corresponding curves in the two cels contain the same number of points. Consequently, at some stage in the interpolation there must be a change in the connectivity of the intermediate cel from the initial to the final. YACAS makes this change at the half-way point in the interval. If the connectivities of the two cels are different, this can result in the abrupt dissapearance of some vectors and the appearance of others. The user can minimize this effect by careful cel design.
(3) If the two pictures are poorly designed, the interpolation can exhibit an effect called "collapsing". This effect occurs when the paths of many of the picture's points pass through or near a single point (often the picture's center). Again, careful design can minimize this effect. (4) "Disintegration" is an effect that can occur when the two pictures have wildly different shapes, or their points do not correspond in spatial position. In this situation the starting picture can lose all semblance of meaningful shape (it disintegrates) before it assumes the shape of the final picture.

Section 3
YACAS Subroutines

The following section lists the subroutines available to the YACAS programmer along with a description of the parameters required and the actions performed. The list is presented in four sub-sections according to a broad functional grouping; the routines are listed alphabetically within sub-section. An understanding of the first two sections of this guide will assist in making good use of these routines.

YACAS opens two files which may be used by the user. One is a print file used by YACAS as a log file to print a summary of the execution of the commands it executes, and error messages if any. This file is declared:

LP (KIND=PRINTER)

The second file is a card file used by READCEL.

CARD (KIND=READER)

NOTE: In the descriptions below, those commands marked with an "*" are not available under the FILMMAKER program (see Appendix 3).

3.1 Operating Environment Commands

CARTOONING(VP, STEP)

Used to control paper-plotter output. VP indicates the number of frames to be plotted across the width of the plotter paper (values may be 1 to 4; default is 3). STEP indicates the number of time units to skip between plotting frames. (Default is 24 FRAMES ... giving one plot per second (every 24th frame)). Plotting begins with the current frame.

* CAPTION(MSG, LEN)

A caption may be added beneath frames output to the

paper-plotter. The character string in MSG (a REAL ARRAY, or quoted string) with length of LEN characters will be written below the next paper-plotted frame. Only one caption per frame will be plotted (the latest).

DUMPIC(PIC)

Causes a formatted dump of the picture description of picture PIC to be printed on the program's log (file "LP"). The picture's header information, and cel description are printed, but not that of any sub-pictures. This command is primarily a debugging tool to be used when the values of specific picture parameters need to be inspected.

* ENVIRONMENT(DEVICE)

This routine initializes the system and indicates the output medium to be used. DEVICE may be "FILM", "PAPER", "BOTH", or "NONE"; anything else is treated as "NONE". "BOTH" means that both FILM and PAPER are used for output. "FILM" means that a film file will be produced. If "FILM" (or "BOTH") is specified, a TITLE must be provided for the file FILM as part of the work-flow control cards for the job. A note is entered in the program log indicating the output device(s) selected.

FILMSPEED(N)

Each frame that is calculated for the film may be duplicated an arbitrary number of times (max 32767). N indicates the number to be used; the default is 1. 12-frame animation may be produced by setting FILMSPEED to 2. Long holds can be most efficiently implemented through use of FILMSPEED.

* FINISHED

This must be the last statement of the film production part of the user program. It must appear in the same

ALGOL block as the ENVIRONMENT statement. One of its side-effects is to close the film file and print a message on the program log (file "LP"), indicating the size of the file.

INHIBIT(ACTION)

ACTION may be "FILM", "PAPER", "BOTH", or "ADVANC". The command causes supression of output to the film file, paper-plotter, or both, or the supression of frame advance (succeeding logical frames appear superimposed on one physical frame).

RESTORE(ACTION)

This command reverses the effect of the INHIBIT command, and may take the same values as arguments.

UNITS(TIMESTEP)

TIMESTEP may be "FRAMES", "SECS" (for Seconds), or any positive value. It represents the time-units used when specifying starting times and durations for the INTERVAL and ROLLIT routines. The minimum time-division is the "frame"; numeric values indicate the number of frames per time-unit. "FRAMES" represents the value 1; "SECS" represents the value 24.

WINDOW(XLL, YLL, XUR, YUR)

The portion of the co-ordinate space visible in a frame is defined as a rectangular area with sides parallel to the axes. The area's lower-left corner is at (XLL, YLL) and its upper-right corner is at (XUR, YUR). The default window is (0,0) and (20,15). This has an aspect ratio of 4:3 which corresponds closely to the normal 16mm film aspect ratio.

3.2 Image Definition Commands

ARC(XS, YS, XE, YE, XC, YC, N, C)

ARC generates calls to VECT to describe a quasi-

circular arc beginning with a point at (XS,YS) and ending at (XE,YE). The arc is produced by rotating a vector about (XC,YC) in ABS(N) steps; the direction of rotation is clockwise for negative N, and counter-clockwise for positive N. The connectivity of the first vector is given by C.

ATTACHTO(N)

The picture currently open is to be attached to its father's attachment point number N. If N = 0, or is invalid, the current picture is attached to the father's pivot point (the default case).

BGNATT

When describing a picture explicitly, BGNATT indicates that succeeding calls to VECT indicate attachment point locations. BGNATT may not be called before BGNOUT for any particular picture.

BGNOUT

Indicates that succeeding calls to VECT describe vectors in the picture's outline. BGNOUT must preceed any call to BGNATT for a given picture.

BRIGHTNESS(N)

Sets the intensity value for the current picture to N. N may have values between -16 and +16; only values +1 to +8 are visible.

CLOSEP

Used to indicate the end of a picture description. There must be a CLOSEP for each OPENP.

COPYPIC(P1, P2)

Creates a picture (P2) as an exact copy of picture P1. This includes a copy of P1's cel (rather than a sharing of its cel).

DELETE(PIC)

Removes picture PIC from the data structure. If PIC's

cel is being used for more than one picture, the cel will be retained. If PIC is part of an articulated picture, all the pictures for which PIC is a root will also be deleted.

INCLUDEPIC(PIC, N)

Adds picture PIC as a son of the currently open picture, attached to the current picture's attachment point #N. PIC must be either a simple picture, or the root of an articulated picture.

PIC := OPENP

Signals the beginning of a picture description. The value returned is the picture number. Picture descriptions may be nested to indicate the hierarchy of an articulated picture. When describing articulated pictures, the cel description of a picture must be completed before any nested OPENPs are issued.

PIVOT(X, Y)

Indicates the position of the pivot point of the current picture's cel. By default, the pivot point is the geometric center of the picture.

POLYGN(SIDES, X, Y, RADIUS)

Calls VECT to create the vectors of a regular polygon with SIDES sides. Each point is RADIUS units from the center of the polygon. The polygon description begins with an invisible vector, and ends at exactly the same point it started with. The starting point is at (X+RADIUS, Y).

READCEL(NAME)

Reads a cel description from the card-image file CARD, generating the appropriate calls to VECT, PIVOT, BGNOUT and BGNATT to describe the cel. The card is free-format; the details are contained in appendix d. NAME is a character string (REAL ARRAY dimensioned at least [0:5]) in which the cel name will be returned (if

present in the set of cards). When using READCEL to describe a picture, calls to VECT may preceed that to READCEL to add more detail to the picture, and may follow READCEL to add attachment points.

USECEL(PIC, X, Y, SX, SY, A)

Makes the currently open picture a new instance of the cel used by picture PIC. The position (X,Y), scale factors (SX,SY), and orientation angle (A) attributes of the new picture may be specified explicitly, or may be copied from those of PIC (when (X,Y) = (0,0), (SX,SY) = (0,0), or A = 0). When USECEL is used, the other picture description commands (VECT, READCEL, etc) may not be issued for the same picture.

VECT(TYPE, X, Y)

Indicates that a point is to be added to the cel description of the currently open picture. The point is connected to the previous point (if there is one) with a visible or invisible vector, depending on the value of TYPE: the connection is visible if type = 0, invisible otherwise. The connection is invisible when there is no previous point (ie. at the start of the Detail or Outline sections. TYPE is ignored when VECT is used to indicate attachment points.

3.3 Motion Commands and Dynamic Functions

Motion commands, except INTERPOLATE, may be issued in static mode (outside an interval description), or in dynamic mode (as part of an interval description). In static mode the motion is performed immediately (between frames). In dynamic mode the motion is queued in the agenda to be performed by ROLLIT. The pictures affected by a motion will be drawn automatically by ROLLIT for each frame of the interval in which they are used. INTERPOLATE may be called only in dynamic mode,

but neither of the pictures used will be automatically drawn (since only the cel is affected).

DISSOLVE(P1, N1, P2, N2)

Change picture P1's intensity from its current value, to N1, and picture P2's intensity from its current value to N2. N1 and N2 may have any value(s) in the range -16 to +16 (only +1 to +8 are visible). Usually this routine is used to dissolve from picture P1 to picture P2 by lowering P1's intensity to 0, while raising P2's intensity to some appropriate value (say 6).

DRAW(PIC) and DRAWF(PIC)

These indicate that picture PIC is to be drawn in the current frame; DRAWF indicates, in static mode only, that this is the last picture to be drawn in the current frame, and that the system is to do an ADVANCE automatically.

FADE(PIC, N)

Changes picture PIC's intensity to N (similar to DISSOLVE except that it acts on only one picture).

INTERPOLATE(PIC1, PIC2)

Indicates that shape (cel description) of picture PIC1 is to be changed to that of picture PIC2. The method used is the standard linear interpolation method described in the "Picture Interpolation" part of section 2. Note that as this routine changes the cel description used by this picture, the shape of any other picture using this same cel will also be changed. If the shapes of other pictures using the same cel are not to be changed, use COPYPIC to create a duplicate cel (in a new picture) then INTERPOLATE that picture.

MOVETO(PIC, X, Y)

Change the position of the picture PIC from its current position to (X,Y).

PAN(DX, DY)

Changes the position of the window by the amount (DX,DY); the size of the window will not be changed.

ROTATETO(PIC, A)

Change picture PIC's orientation to A (degrees) from its current orientation. The picture will end up oriented at A degrees after "spinning" through (A - <current orientation>) degrees. Note that orientation is specified as, and stored as the full (not modulo 360) value.

SCALEBY(PIC, SX, SY)

Change the scaling factors for picture PIC by (SX,SY). The new factors will be ((initial X-value)*SX, (initial Y-value)*SY).

ZOOM(SCALE)

Changes the size of the window by the inverse of the factor SCALE. ZOOM changes the apparent size of all pictures within the window by the factor SCALE (just as PAN changes the apparent position of all pictures within the window by (-DX,-DY)).

Dynamics Functions

These functions may only be issued as part of an interval description and must immediately follow the motion command they affect.

CUSHIONED

A dynamics function: a motion is to be carried out with cushioned dynamics. The motion will start slowly, speed up to a nearly constant speed, then slow down towards the end of the interval.

LINEAR

A dynamics function: a motion is to be carried out with linear dynamics: the rate of change of the

affected function is to be constant from frame to frame in the current interval.

SINUSOID(PERIOD)

A dynamics function: a motion is to be carried out with sinusoidal dynamics. The motion starts slowly, speeds up, then slows down at the motion's extreme, speeds up again, then slows down again at the initial "position" (size, shape, orientation, etc) at the end of one full period. The parameter PERIOD indicates the number of full periods to be used in the current interval. PERIOD may have any non-zero value (including fractional ones).

3.4 Miscellaneous

ADVANCE

Indicates that the current frame has been completed, and that the description of the next frame is to begin. It will usually cause a physical frame advance (unless previously "INHIBIT"ed).

INTERVAL(T1, T2)

This command marks the beginning of a scene description. All function commands issued during an interval will be carried out over that interval. An interval is terminated by either another INTERVAL, a ROLLIT, or a SETSTATIC command. The interval starts T1 time units from the current time, and lasts for T2 time units. T1 may have any non-negative value (0 represents "now" - the current time), or may be "*", also representing "now". T2 may have any positive, non-zero value, representing the duration of the interval in the current time units. It may have a fractional value, but the minimum duration will be 1 frame.

ROLLIT(T1, T2)

Indicates that one or more scene descriptions have now

been completed, and that the indicated portion of the agenda is to be processed. T1 is the starting "time" in current units from "now"; T2 is the duration to be processed. T1 may have any non-negative value, or the literal "*" (indicating "NOW"); T2 may have any positive non-zero value or the literal "*" (indicating "everything in the agenda"). If T1 is greater than 0, that portion of the agenda preceeding T1 will be processed, but no frames will be produced; if the duration extends past the end of the time indicated in the agenda, blank frames will be produced for the extra time; if there are gaps in the agenda for the interval (T1,T2) blank frames will be produced for those gaps. A note will appear in the program execution log (file "LP") for each call to ROLLIT, indicating the frames produced.

SETSTATIC

May be used to indicate the end of a scene description; places the program into static mode in which actions (such as MOVETO or SCALEBY) operate immediately rather than over a time interval. Normally, a call to INTERVAL, or to ROLLIT would be used to terminate a scene description.

Section 4
PLABAK

YACAS is composed of two distinct components: a set of subroutines (described in section 3) to be used in a user-written program to produce a "film" as a file of information, and a program (PLABAK) to display the film on a graphic display unit for review and filming. This part of the User Guide describes the facilities and features of PLABAK, and discusses the manner of interaction with the program. Appendix 2 goes into specific detail concerning file names, disk pack identification etc concerned with the actual running of the program as at the time of writing. The appendix will also describe methods to be followed to transfer the film files from the B6700 to the GT44 for use by PLABAK.

The purpose of PLABAK is to display "films" from film files on the graphic display device (the GT44) for purposes of review and filming. It allows the user to have some control over the playback process: specifically, the speed of display can be controlled in a rather crude way, the film file can be partitioned into "scenes" for separate display, and some degree of control over the display window is provided. These control features are described more fully in section 4.2.

PLABAK is a program written for the PDP-11 GT44 computer system in the PDP-11 MACRO assembly language. It is written to run under the RT-11 operating system. Some familiarity with the use of the GT44 and RT-11 is assummed in the following discussion. Appendix 2 contains enough information to enable an inexperienced person to run PLABAK after a short instruction period from an experienced RT-11 user.

4.1 Running PLABAK

This section describes the over-all method of running the PLABAK program, and discusses the over-all syntax of the actions the user may take. Section 4.2 goes into more specific detail concerning actions the user may make and their effects.

When PLABAK is run, it asks the user to name a film file to be displayed. When the user does this, PLABAK locates the file, gets set to begin, then asks the user for "permission" to proceed. When this is given, PLABAK displays the film from beginning to end, then asks the user whether it should re-display the same file, or a new one. When the file to be displayed has been established, it repeats the process from the "permission-to-proceed" step.

To run the PLABAK program, ensure that the correct disk is mounted as the system disk, then issue the RT-11 command:

```
.R PLABAK
```

The program will respond with a title line, then a request for the name of a file to be displayed.

```
PLABAK:  WHICH FILM (FILE)? *
```

At this stage, enter the name of the film file to be displayed. PLABAK assumes the file extention "FLM". If the file cannot be found, or some other similar error occurs when trying to locate the file, a suitable message is printed and the request for the file name is repeated. When the file has been found and is set to go, PLABAK will request "permission" with:

```
*** BEGIN?  ***
```

Type the carriage-return key (<CR>) to start the display.

When the end of the film is reached, PLABAK prints a message giving the number of frames in the film, then asks if it should repeat the film:

```
SCENE ENDED AT FRAME nnn.
REPEAT FILM?  (/Y) *
```

You may respond with "/Y" to repeat the same film, "/E" to exit (return to RT-11), or with the name of a new file to be displayed. If either the film is to be repeated, or a new film is to be displayed, the process repeats from the "*** BEGIN ***" step. If the new film named is not found, an appropriate message is printed, and the process begins again at the "WHICH FILM (FILE)? *" step.

Interrupting PLABAK

At any time while a film is being displayed, the user may stop the display by typing a carriage-return. PLABAK will stop processing the film file as the end of the current frame, display this frame, type an "*" on the console, and pause. The display may be re-started by typing a second carriage-return. A film may be interrupted as often as desired.

Terminating PLABAK

PLABAK may be terminated in any of three ways. The user may type control-C (^C) at any time to return to the RT-11 monitor. At any time that PLABAK is waiting for a response from the user (eg. waiting for a file to be named, waiting to BEGIN, or has been interrupted), "/E" may be typed to cancel the display of the current film and return to the RT-11 monitor. If the display of a film has been interrupted, "/T" may be typed to terminate the display. PLABAK retains control and replies with the "REPEAT FILM ?" request. "/E" is the preferred way to return to the monitor.

4.2 Interruption Commands

At any time that PLABAK is waiting for a response from the user, one or more commands may be entered. The commands allow the user to control the playback display rate, request a fixed display window, or select specified sub-sections of

the film for display. A "HELP" command provides brief instructions for operating the program, and a list of the available commands.

The commands are implemented using the RT-11 switch syntax. A switch consists of a slash ("/") followed by a single alphabetic character, optionally followed by one or more values. Values consist of a radix-code character (":" for Octal and "!" for Decimal values) followed by 0 or more numeric characters. Values may be positive or negative.

```
eg:  /X        - switch with no values
     /A:1      - switch with value 1 (octal)
     /B!-9     - switch with value -9 (decimal)
     /C!8:10   - switch with two values
                       (8 decimal and 10 octal)
```

The "/H" switch is the HELP command. A copy of the listing it produces appears as Figure A.4-1. A description of each of the available commands follows.

```
PLABAK:  WHICH FILM (FILE)? */H

** PLABAK **
TO NAME THE FILE TO BE USED, TYPE:
        <FILENAME><CR>
  (DEFAULT FILE EXTENTION IS 'FLM').
PROCESSING MAY BE INTERRUPTED AT ANY TIME BY
TYPING A <CR>.  THE FOLLOWING SWITCHES MAY BE
USED IN RESPONSE TO THE '*' PROMPT:
  /E ... EXIT: RETURN TO MONITOR
  /H ... HELP: THESE INSTRUCTIONS
  /T ... TERMINATE FILM
  /V ... TURN VIEWPORT FRAME ON/OFF
  /W ... SPECIFY A FIXED WINDOW
  /Z ... TURN FOCUS IMAGE ON/OFF
  /Q:N ... SET DISPLAY SPEED:
             N TICKS/FRAME; PAUSE IF N = 0.
  /S:N ... SET FRAME SELECTION CODE:
             +N - DISPLAY EACH N TIMES
             -N - DISPLAY EVERY NTH FRAME
  /B:N:M .. SPECIFY A FILM SEGMENT:
             BEGIN AT FRAME N; RUN FOR M FRAMES.
PROCESSING MAY BE RESUMED BY TYPING <CR>.
NOTE: NUMBERS TYPED ARE OCTAL (:N) OR DECIMAL (!N)
```

Fig. A.4-1
PLABAK '/H' Listing

PLABAK Control Commands.

/E ... End processing and return to RT-11 monitor.

/H ... HELP command; produces the listing shown as Figure A.4-1.

/T ... Terminate processing of the current film but retain PLABAK control.

/V ... The viewport is a rectangular area on the display screen in which the film is displayed. It appears on the screen as a rectangular box. Outside the viewport is a frame counter showing the frame number being displayed, and possibly a "RUNNING SLOW" message if PLABAK cannot maintain the desired display rate. This viewport may be turned off and on by using the "/V" switch.

/W ... The user may specify that a fixed window is to be used for display of a film. This fixed window will be used in place of that specified by the film file (which will appear as a rectangular box of dashed lines whenever it falls within the fixed window). PLABAK will prompt for the lower-left co-ordinates then the upper-right ones for the fixed window. The specified window may be cancelled by entering a second "/W" command.

/Z ... A special image consisting of crosses at the corners, middle of the sides, and center of the viewport may be turned on and off with the /Z switch. This image can be used to align a camera prior to filming. The image includes an intensity scale; this should be adjusted with the display's intensity adjusting control, so that the bar at intensity 1 is just barely visible.

/Q:n .. There are two ways to control the speed at which the film is displayed; this switch and the /S switch. The /Q switch controls the rate at which frames are displayed. The value 'n' is the number of clock

ticks (1/50th of a second) the frame is to be allowed for display. If 'n' is 0, each frame is displayed until a carriage-return is typed on the console. The normal value for n is 2; typical values would be 0<=n<=10. Note that -1 is treated as 65535.

/S:n .. The second manner by which film speed can be controlled is accomplished by frame selection. The speed can be slowed by displaying each frame some number of times; it can be speeded up by skipping (not displaying) frames. The value of n accomplishes this: when n is positive, each frame will be displayed n times. When n is negative, the absolute value of n is used: n-1 frames will be skipped resulting in the display of every n'th frame.

ie: For /S:2, each frame will be displayed two times resulting in the speed being halved.

For /S:-2, every second frame will be displayed resulting in a doubling of the speed.

/B:n:m . Portions of a film file may be selected for display through use of this switch. The n value indicates a beginning frame number (the frame at which the sequence is to begin) while the m value indicates the duration (number of frames) in the portion (the number actually displayed depends on /S value). The beginning frame number should be greater than the current frame (displayed above the upper-left corner of the viewport). Values of 0 for n or m set "undefined" values - the film will be displayed from the current frame (for n=0) until the end (for m=0) - n=0 and m=0 are the initial values. Once the /B values have been set and the portion displayed, PLABAK will display no more of the film until the /B values are set to new values.

Section 5
Extending YACAS

YACAS, in its present form is a rather minimal system. For this reason, and because users may wish to have more specialized routines available, YACAS provides facilities to aid the user in adding new routines to the set supplied. Additional routines of three types may be supplied by the user: static routines such as picture describing routines (eg POLYGN), motion routines (such as MOVETO), and dynamics functions (such as CUSHIONED). The following sections describe the techniques to be used. Casual users are advised not to attempt extensions to the system until some experience has been gained.

Note that no direct method is provided to make permanent changes to YACAS. The extensions described here are temporary ones that users may make for their own programs.

5.1 Manipulating the Data Structure

User extensions to YACAS will normally be expected to manipulate the data structure; several commands are available to facilitate this. The basic methods of picture storage, agenda storage and processing, and film file creation are not available for the user to alter. The user may, however, inspect and alter most information in these data structures (except the film file).

Pictures may be created in the usual way (via OPENP, CLOSEP, VECT etc) and deleted (via DELETE) at anytime. Picture header information may be inspected or altered via the PICINFO command. Cel information may be inspected or altered via the CELINFO command. Articulated picture structures may be traced through careful use of the

RELATIONS command. Values stored in the agenda may not be inspected or altered. The user is restricted to adding new entries via the MARKAGENDA command, or adding dynamics via the DYNAMICS command. Discussion of the routines PICINFO, CELINFO, RELATIONS, MARKAGENDA and DYNAMICS may be found in section 5.5.

Certain global variables may be of use to users wishing to extend YACAS. The values of these variables should not be altered by the user.

DYNAMICSW - TRUE indicates dynamic mode, else static mode.

ANIMSW - TRUE indicates that the agenda is being processed.

CELOPN - TRUE when a picture has been OPENPed and VECT may be used to add vectors to its cel description.

INTRVLSTART - Starting frame number for current interval.

INTRVLEND - Ending frame number for current interval.

TIMEUNITS - Number of frames per time unit selected by user (TIMEUNITS = 1 for "FRAMES", = 24 for "SECS").

5.2 Adding Static Routines

Static routines are the simplest to add since they are called only by the user's program. All of the commands of YACAS are available to create, delete, and alter pictures. The user's program can determine whether an INTERVAL is in force through the global boolean variable DYNAMICSW (it will be TRUE). The user's program can determine whether a cel description has been started through the boolean global variable CELOPN (it will be TRUE). Since user-written static routines are no different than any other user-written routines (except, perhaps, in intent) the methods required to add and make use of them are the same as for the subroutines of a normal user program.

Example: BOX Routine

The following routine will add the vectors for a rectangular box, as a separate curve, to the currently open picture.

```
PROCEDURE BOX (XC, YC, LENGTH, WIDTH);
  VALUE XC, YC, LENGTH, WIDTH;
  REAL  XC, YC, LENGTH, WIDTH;
COMMENT
Add the vectors for a rectangular box to the
currently open picture.  XC, YC is the center of
the box while LENGTH is its size in the X direc-
tion and WIDTH is its size in the Y direction;
IF NOT CELOPN
 THEN  WRITE(LP,<"CELOPN - CEL NOT OPEN; BOX NOT",
                                    " CREATED.">)
 ELSE BEGIN
  REAL X , Y;
  X := XC - LENGTH *0.5;
  Y := YC - WIDTH  *0.5;
  VECT(1,X,Y); VECT(0,X+LENGTH,Y);
  VECT(0,X+LENGTH,Y+WIDTH);
  VECT(0,X,Y+WIDTH); VECT(0,X,Y);
END BOX;
```

5.3 Adding Motion Routines

For the user to add a motion command to the set, two actual routines need to be prepared. One routine is the form called directly by the user; its function, normally, is to place an entry in the agenda indicating which motion is to be carried out when the agenda is processed. The second routine is that which ROLLIT calls when it processes the agenda. Since ROLLIT must know the names of all the routines it may call at compile time, the names of the user-added motion routines are pre-determined (see Table A.5-1).

5.3.1 Motion Routine Conventions

Since ALGOL provides no mechanism for storing a reference to a procedure in an array, YACAS uses a symbolic technique. Numbers are associated with each motion routine (including user-written ones) to be stored in the agenda. This number is later used by ROLLIT to select the correct routine to call (ROLLIT uses a CASE statement). These

numbers also have symbolic names (also shown in Table A.5-1).

Finally, the YACAS source file contains dummy routines for each of the possible user-written ones (to resolve the references to the routines in ROLLIT). For the user to add his own routines, the dummy ones must be removed. The YACAS source file has been organized in such a way that setting an ALGOL "User $-option" will remove the necessary code (one $-option for each dummy routine). These names are also shown in the table.

Program Name	Symbolic number	$-option
ANIMUSER1	AGUSER1	ANIMUSER1
ANIMUSER2	AGUSER2	ANIMUSER2
ANIMUSER3	AGUSER3	ANIMUSER3

Table A.5-1
Symbolic names for user motion routines.

The routine called directly by the user has the primary function of making an entry in the agenda showing which dynamic routine is to be called, with what parameters, and for what interval (provided it is called in dynamic mode). The routine can have a secondary function of carrying out the action immediately. The example in section 5.3.3 illustrates both primary and secondary functions.

In carrying out its primary function, the routine called by the user typically translates the parameters passed to it into the desired form for the routine MARKAGENDA which makes an entry in the agenda (see section 5.5 for a description). The parameters of MARKAGENDA provide for up to 4 user parameters, a picture number, and the starting and ending frame numbers for the INTERVAL (available as the globals INTRVLSTART and INTRVLEND respectively).

The user-written routine to be called by ROLLIT must begin with a standard-format procedure declaration:

```
PROCEDURE ANIMUSER1 (PIC, PARM1, PARM2, PARM3, PARM4,
                                    F0, F1, T, T1, T2);
  VALUE   PIC, PARM1, PARM2, F0, F1, T, T1, T2;
  REAL    PARM1, PARM2, PARM3, PARM4, F0, F1;
  INTEGER PIC, T, T1, T2;
```

The parameters of this declaration will contain the following values when called by ROLLIT:

PIC - The picture number entered in the agenda by MARKAGENDA.

PARM1 to PARM4 - The user parameter values stored in the agenda.

F0 and F1 - The preceding and current dynamics function values (see section 5.3.3).

T, T1, T2 - The current, interval starting, and interval ending frame numbers.

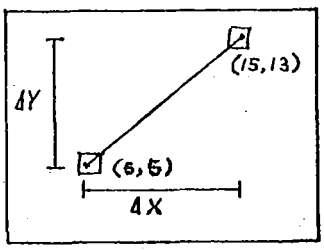
The user's procedure will typically make changes to the picture, or perhaps to the picture's cel, using the parameters and function values passed to it. Provided the changes are made via the PICINFO or CELINFO commands, the changes will appear when PIC is automatically drawn by ROLLIT. If any changes are made to the Picture Header Table or to the Cel Table directly, then the flag bits PICCOPIED(PIC) and/or CELCOPIED(HDRLNK(PIC)) must be set to FALSE.

Parameters PARM3 and PARM4 have a special property. Their values may be altered by the user's program and these new values are the ones that will be returned when the routine is next called by ROLLIT.

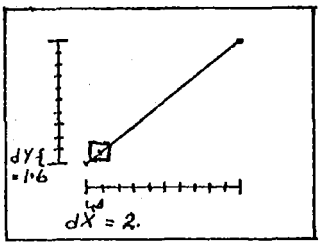
5.3.2 Motion Calculations

The technique used within YACAS to achieve motion, is to calculate the amount of change needed to get from the current state to the final state in the remaining number of frames. This process may be illustrated by considering the operation of the MOVETO command in conjunction with Figure

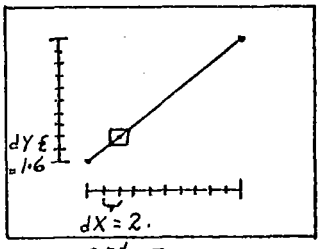
A.5-1.



a. MOVETO Example



b. 1st FRAME



c. 2nd FRAME

Fig. A.5-1
A MOVETO example.

In this example, the picture BOX is to move from (5,5) to (15,13) in 5 frames. For the first frame of the sequence, the picture must be moved 1/5 of the distance involved (Fig. A.5-1b).

ie: (15-5) * 1/5 = 2 units in the X direction

and (13-5) * 1/5 = 1.6 units in the Y direction.

The picture's new position is thus (7, 6.6). For the next frame, there are 4 frames left in the interval, so the picture must be moved 1/4 of the distance (Fig. A.5-1c). The movement required is

(15-6) * 1/4 = 2 in the x direction

and (13-5.8) * 1/4 = 1.6 in the y direction.

Thus the picture's position for the second frame of the sequence is (9, 8.2). This procedure continues for the whole sequence.

Using this procedure, the increment of change from one frame to the next is constant (in the example, there is a 2 unit change in the X component, and 1.6 units change in the Y component). The dynamics of the motion will be Linear. To achieve Cushioned dynamics, the amount of change must vary; the amount of change must be small at the start,

larger in the middle, and small again at the end of the interval (Fig. A.5-2).

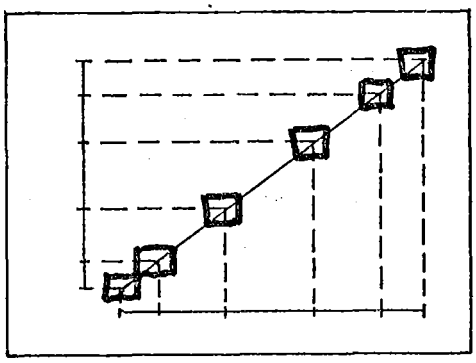


Fig. A.5-2
MOVETO with CUSHIONED dynamics.

YACAS achieves this effect through use of the dynamics functions. Dynamics functions return a value in the (inclusive) range 0 to 1. This value represents the proportional amount of change required to change a picture from its original state to current intermediate state given the desired final state. The LINEAR dynamics function simply returns the value of

F1 = (Current frame)/(Total frames)

The CUSHIONED dynamics function returns a value calculated using the LINEAR value and the COSINE function to achieve the suitable non-linearity. The formula used is:

Fc = 1/2 + 1/2 * COS(F1*<pi>+<pi>).

YACAS cannot use the dynamics function values directly for its calculations because the "original" position is lost after the first frame. A few algebraic manipulations can be used to show that the amount of change required may be calculated as a function of the current situation (position in the case of the MOVETO routine; angle in the case of the ROTATETO routine), the final situation, the previous function value and the current function value. The derivation will not be described here; only results

produced.

In the MOVETO example, the formula that results, giving the new position is

X' = X" - (X" - X) * F

and Y' = Y" - (X" - X) * F

where F = (1 - F1) / (1 - F0)

and (X, Y) is the previous position

(X', Y') is the new position

(X", Y") is the ultimate position

F0 is the previous dynamics function value

F1 is the current dynamics function value.

This formula is not a general one. It works for the MOVETO and ROTATETO situations (an equivalent formula can be used to calculate the intermediate orientations), but fails if applied to the SCALEBY command, and fails if applied to the example discussed in section 5.3.3 (the SHEAR example). To illustrate the technique used to implement motions in YACAS, the formula used to determine the postion of the intermediate points in the SHEAR command is derived below. A description of the SHEAR command appears in section 5.3.3; a brief outline appears here.

The SHEAR command alters the shape of a picture's cel by displacing the positions of the cel's points. Only the X component of each point is displaced; the amount depends on the magnitude of the Y component, and the Tangent of a stated angle (see fig. A.5-3). For the static transformation the formula used is:

X' = X + Y * TAN(ANGLE) ... (1)

(Y' = Y)

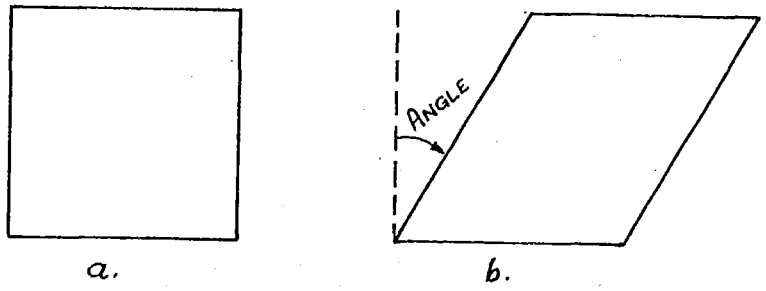


Fig. A.5-3
The SHEAR routine: (a) before, (b) after

The derivation of the formula to be used for the animating part of SHEAR can be shown in the following steps (using with a 4 step example - fig. A.5-4).

(1) The initial situation has a point at (X,Y), and is to be SHEARED through ANGLE in 4 steps.

(2) The first frame of the sequence can calculate the new X value (X') as

X' = X + Y *TAN(ANGLE * 1/4) ... (1)

(3) The second frame, using the static calculation formula, will result in the X value (X"):

X" = X + Y * TAN(ANGLE * 2/4) ... (2)

(4) However, for the animation case, we no longer know what the value of X was, but do know X'. Manipulating (1) we can compute X as:

X = X' - Y * TAN(ANGLE * 1/4) ... (3)

(5) Substituting into (2) and simplifying yeilds:

X" = X' + Y * (TAN(ANGLE * 2/4)
- TAN(ANGLE * 1/4)) ... (4)

(6) Inspecting (4) we note that the values 1/4 and 2/4 are just the values that would be returned by the LINEAR dynamics function for frames 1 and 2 of the 4 frame sequence; we will now call them F1 and F2 respectively. Now:

X" = X' + Y * (TAN(ANGLE * F2)
- TAN(ANGLE * F1)) ... (5)

(7) Generalizing, the formula may be applied for any step by using F1 as the dynamics function value from the

previous frame of the sequence (it will be 0 if this is the first frame), F2 as the current function value, X' as the X value calculated for the previous frame, and X" as the current value being calculated. The increment of change for each frame of the sequence is thus given by

Y * (TAN(ANGLE * F2) - TAN(ANGLE * F1))

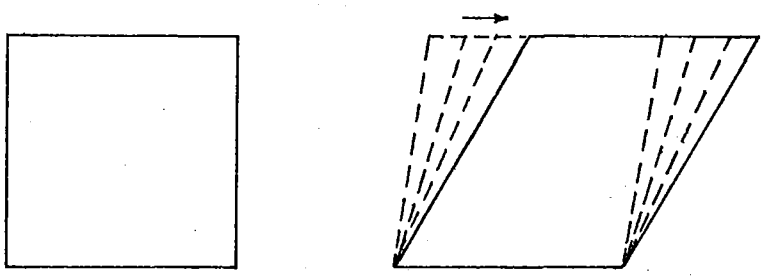


Fig. A.5-4
SHEAR as a motion.

5.3.3 An Example: The SHEAR Routine

Figures A.5-5 and A.5-6 show the coding for the two parts of a new motion command. Figure A.5-5 shows the coding for the routine that the user would call, while Figure A.5-6 shows the code for the routine to be called by ROLLIT. Figure A.5-7 shows a simple program using the new command.

The SHEAR routine transforms a picture's cel by displacing each point of the cel in the X direction by an amount proportional to its Y value. The constant of proportionality is the Tangent of the specified angle. Note that the angle is restricted to a value in the range -60 to +60 degrees.

The SHEAR routine shown will operate on the cel of the indicated picture only. If the picture is articulated, the cels of other pictures in the structure will not be affected. To adapt the routines to operate on all cels in

the structure, additional loops would be needed to sort through the structure (via the RELATIONS command).

```
PROCEDURE SHEAR ( PIC, ANGLE );
  VALUE  PIC, ANGLE;
  INTEGER PIC;  REAL ANGLE;
COMMENT
   SHEAR transforms a picture's cel by altering the X
position of each point in the picture by an amount
proportional to the product of the point's Y value, and the
tangent of the supplied ANGLE.
ANGLE is allowed values of -60 to +60 (degrees);
IF ABS(ANGLE) > 60.  THEN      % CHECK FOR LEGAL ANGLE
    WRITE(LP,<"SHEAR - ILLEGAL ANGLE(",F8.3,")">,ANGLE)
 ELSE IF DYNAMICSW THEN        % PUT ENTRY IN THE AGENDA
    MARKAGENDA( AGUSER1, PIC, ANGLE, 0., 0., 0.,
                              INTRVLSTART, INTRVLEND )
  ELSE BEGIN                  % ELSE: DO THE SHEAR NOW
      REAL  SECTION, CURVE, POINT, X, Y, TANGLE;
      TANGLE := TAN(ANGLE * 0.01745329);
      FOR SECTION := 0,1,2  DO BEGIN
        CURVE := 1;
        WHILE  CURVE > 0   DO BEGIN
          POINT := 1;
          CELINFO(PIC,1,SECTION,CURVE,POINT,X,Y);
          WHILE  POINT > 0   DO BEGIN
            X := X + Y * TANGLE;
            CELINFO(PIC,0,SECTION,CURVE,POINT,X,Y);
            POINT := POINT+1;
            CELINFO(PIC,0,SECTION,CURVE,POINT,X,Y);
            END;
          CURVE := CURVE+1;
          END;
        END;
END OF SHEAR;
```

Fig. A.5-5
The SHEAR command: user entry-point.

```
PROCEDURE ANIMUSER1 (PIC, ANGLE, P2, P3, P4,
                                    F1, F2, T, T1, T2);
  VALUE PIC, ANGLE, P2, F1, F2, T, T1, T2;
  REAL  ANGLE, P2, P3, P4, F1, F2;
  INTEGER  PIC, T, T1, T2;
COMMENT
This routine "animates" the SHEAR command.  It is called by
ROLLIT to calculate the increment of SHEAR to picture PIC in
response to an entry in the agenda;
IF  ANIMSW  THEN BEGIN      % ENSURE CALLED PROPERLY
   REAL  SECTION, CURVE, POINT, X, Y, F;
   F := TAN(ANGLE * F2 * 0.01745329)
                    - TAN(ANGLE * F1 * 0.01745329);
   FOR  SECTION := 0,1,2   DO BEGIN
     CURVE := 1;
     WHILE  CURVE > 0.  DO BEGIN
       POINT := 1.;
       CELINFO(PIC,1,SECTION,CURVE,POINT,X,Y);
       WHILE  POINT > 0.  DO BEGIN
         X := X + Y * F;
         CELINFO(PIC,0,SECTION,CURVE,POINT,X,Y);
         POINT := POINT + 1;
         CELINFO(PIC,1,SECTION,CURVE,POINT,X,Y);
         END;
       CURVE := CURVE + 1;
       END;
     END;
END OF ANIMUSER1;
```

Fig. A.5-6
The SHEAR command: ROLLIT entry-point.

```
BEGIN                    % A SHEAR  EXAMPLE
$ SET ANIMUSER1
$ INCLUDE YACASTEXT
PROCEDURE SHEAR( PIC, ANGLE );
  <code as in Figure A.5-5>
  .
  .
  .
PROCEDURE ANIMUSER1(PIC,ANGLE, ......... );
  <code as in Figure A.5-6>
  .
  .
  .
COMMENT
A trivial program to show the use of the user-written new
YACAS command SHEAR;

   INTEGER  PIC;

   ENVIRONMENT("PAPER"); UNITS("FRAMES");
   CARTOONING(2,1);     INHIBIT("ADVANC");
   PIC := OPENP;
     POLYGN(3,10.,7.5,5.);    % PICTURE:
                              %  A TRIANGLE AT (10,7.5)
     CLOSEP;
   DRAWF( PIC );
   INTERVAL( 0., 2.);
     SHEAR( PIC, 30.);
   ROLLIT( "*", "*" );
   RESTORE("ADVANC");
   FINISHED;
END OF MAIN.
```

FIG. A.5-7
A trivial example using the user-written SHEAR command.

5.4 Adding Dynamics Functions

In a manner similar to that required to add motion commands to YACAS, dynamics functions require two procedures to be written - one to be called by the user which marks the agenda appropriately, and one that will be called by ROLLIT. Those routines that will be called directly by the user may have any name and parameters the user wishes; those that will be called by ROLLIT will need to have a specific name and be coded to accept a specific set of parameters as discussed below.

5.4.1 Dynamics Function Conventions

As with user-added motion functions, the user may add up to three new dynamics functions. The standard names for the user-written subroutine, the identifying numeric code, and the ALGOL user $-option to be used in adding the functions are shown in Table A.5-2. An example is discussed in section 5.4.3.

Function name	Symbolic number	$-option
FUSER1	DFUSER1	FUSER1
FUSER2	DFUSER2	FUSER2
FUSER3	DFUSER3	FUSER3

Table A.5-2
Symbolic names for user dynamics functions.

The routine the user writes to mark the agenda needs to do little more than call the procedure DYNAMICS (see section 5.5 for a description). A single parameter may be passed to the dynamic function via the agenda (the example in 5.4.3 does not pass a parameter).

User-written dynamics functions must start with a procedure header of the form:

```
REAL PROCEDURE FUSER (T, T1, T2);
VALUE T, T1, T2; INTEGER T, T1, T2;
```

The parameters that will be passed to the routine are:

T - The current frame number.

T1 - The starting frame number of the interval.

T2 - The ending frame number of the interval. In all cases T1 <= T <= T2.

If the dynamics function is passed a parameter, that parameter's value is available in the global variable DFUNGLOBAL. The user may change the value of this variable and this new value will be made available when the function is next called by ROLLIT.

5.4.2 Dynamics Function Calculations

Dynamics functions should return a value in the range 0. <= Value <= 1. Ordinary functions would normally return the value 1.0 when T = T2. The value returned can be thought to represent the proportion of change from the starting state to get to the current state (at time T). For linear dynamics, the formula used is simply

```
F = (T-(T1-1)) / (T2-(T1-1))
```

Figures A.2-9, A.2-10 and A.2-11 show graphs of the values returned by each of the implemented dynamics functions versus the frame number.

Dynamics functions may be continuously increasing functions (such as LINEAR and CUSHIONED). They may be periodic (such as SINUSOID), or they may be step functions (returning only values of 1 or 0). The calculation used for the SINUSOID function will serve to illustrate a periodic function; DFUNGLOBAL contains a number representing the number of whole periods to fit into the interval:

```
F := SIN(DFUNGLOBAL * (T+1-T1)/(T2+1-T1)
                    - 3.14159/2) * 0.5 + 0.5;
```

5.4.3 An Example: The ACCELERATE Function

The example that follows shows an implementation of an ACCELERATE function. The dynamics are exactly half those of the CUSHIONED dynamics - the change provided starts slowly (change in value is small) and increases (change in value gets larger) towards the end of the interval. A graph similar to that shown in Figure A.2-9 is shown in Fig. A.5-8. The calculation is based on appropriately transforming the quarter-cosine between 0 and 90 degrees.

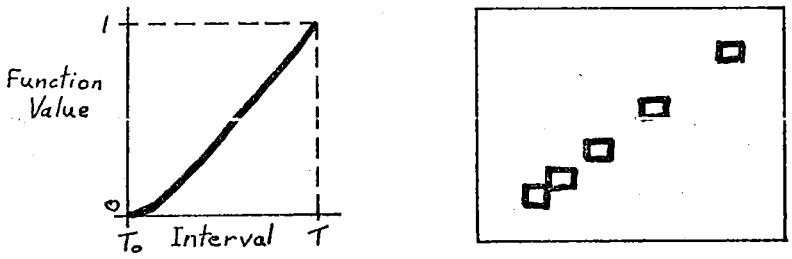


Fig. A.5-8
ACCELERATE dynamics function
and sample results.

The code for the user-called routine is:

```
PROCEDURE ACCELERATE;
COMMENT
     A user-called dynamics function to provide
     accelerate dynamics.  Changes start small and get
     larger as the interval proceeds;
DYNAMICS(DFUSER1,0.);
```

(The user could save the cosmetics of a separate ACCELERATE procedure by simply calling DYNAMICS directly, or by creating a DEFINE to do so).

The code for the actual dynamics function is:

```
REAL PROCEDURE FUSER1 (T,T1,T2);
VALUE T,T1,T2; INTEGER T,T1,T2;
COMMENT
     This function inplements the ACCELERATE dynamics
     function.  It does so by transforming the cosine
     function from 0 to 90 degrees appropriately;
FUSER1 := 1.0 - COS(1.570296 * (T,T1,1)/(T2,T1,1));
```

Finally, to patch the new function into the YACAS system and use it:

```
BEGIN                % TRIVIAL USER-DYNAMICS EXAMPLE
 $ SET FUSER1
 $ INCLUDE  YACASTEXT
PROCEDURE ACCELERATE;
  <procedure body as above>
  .
  .
  .
REAL PROCEDURE FUSER1 (T,T1,T2);
  <procedure body as above>
  .
  .
  .
                % MAIN PROGRAM CODE
 INTEGER PIC;
 ENVIRONMENT ("PAPER"); UNITS ("SECS");
 INHIBIT ("ADVANC");    CARTOONING (2,1);
 PIC := OPENP;    POLYGN (5,5.,5.,2.);    CLOSEP;
 INTERVAL (0,1);
 MOVETO (PIC, 15., 13.); ACCELERATE;
 ROLLIT (0,"*");
 RESTORE ("ADVANC");
 FINISHED;
END of MAIN.
```

5.5 Supplementary YACAS Commands

The following are commands that the YACAS user making extensions to the system may find useful.

DYNAMICS (FUNCTION, PARAM)

Modifies the most recent AGENDA entry to show that the dynamics indicated by FUNCTION (the function number) is to be used. PARAM is the parameter to be stored for the routine. Valid (symbolic) function numbers are DFUSER1, DFUSER2 and DFUSER3.

CELINFO (PIC, TYPE, SECTION, CURVE, POINT, X, Y)

Used to change the values of a cel point, or to retrieve the values. The point is indicated by giving the associated picture number (in PIC), the section (may be 0 or "DETAIL", 1 or "OUTLIN", 2 or "ATTACH"), the curve number and the point number. TYPE indicates whether a value is to be changed (TYPE=0) or retrieved (TYPE=1). If the point does not exist, the value -1 will be returned in POINT; if the curve does not exist, the value -1 will be returned in CURVE and POINT. If CURVE=0, the number of curves in the section will be returned in X. If POINT=0, the number of points in the curve will be returned in X (unless TYPE=0).

MARKAGENDA (ROUTINE,PIC,PARM1,PARM2,PARM3,PARM4,FS,FF)

Makes an entry in the agenda. ROUTINE is a numeric code for the routine to be used; symbolic values that should be used are AGUSER1, AGUSER2 and AGUSER3. PIC is usually the picture number of the picture to be used by the routine. This picture will automatically be drawn in each frame by ROLLIT. PARM1 and PARM2 are user-defined values which will not be changed; PARM3 and PARM4 are user-defined values which may be altered by the user's animation routine (but need not be); FS and FF are the starting and ending frames of the

interval the routine is to be active over. Usually the global variables INTRVLSTART and INTRVLEND will be used.

PICINFO (PIC, TYPE, X, Y, SX, SY, R, I)

A dual-purpose function: when TYPE=0, PICINFO will set picture header information for picture PIC (the values in X, Y, SX, SY, R, and I). When TYPE=1, the values currently in PIC's header will be returned.

X,Y - is the current position of the picture.

SX,SY - are the X and Y scaling factors.

R - is the orientation angle (in degrees).

I - is the picture's intensity value.

RELATIONS (PIC, FATHER, SON, BROTHER)

This routine returns the picture numbers of PIC's father, son and brother pictures. A value of 0 for any (or all) of these items indicates the absence of the relation: a PIC with a non-zero "son" value, but zero for the "father" is a root picture in an articulated set.

VALIDPIC (PIC)

A Boolean procedure that returns the value TRUE if PIC is the number of an existing picture, and FALSE otherwise.

Appendix 1 Using YACAS

This appendix describes the steps to be followed to make use of the YACAS subroutines from a Work-Flow language (WFL) point of view. File names are indicated along with suggested techniques for saving and using files. The information is correct as at the time of publication of this User Guide. Some of these specific details may change. The beginning user is cautioned to check the correctness of this material before using it.

Files

YACAS subroutines exist in source form in the file

(COSC023)YACASTEXT

This file is stored on magnetic tape B172. It is recommended that users copy this file to their own account and save it on tape for future use.

To retrieve the YACASTEXT file and save it on one's own tape (eg. A111):

```
? COPY (COSC023)YACASTEXT AS YACASTEXT FROM B172
? COPY YACASTEXT TO A111
```

The following examples assume that the file has been saved on tape A111 under the user's own account. The examples also assume that the user's program is in the form of a card deck.

To compile and run a program to make a film the following WFL statements may be used:

```
? IF FILE YACASTEXT ISNT PRESENT THEN
        COPY YACASTEXT FROM A111
? COMPILE USEREXAMPLE1 ALGOL
? FILE FILM (TITLE = USERFILM)
? DATA
BEGIN
 $ INCLUDE YACASTEXT
     :
     :
     :   <user program>
     :
     :
```

```
END.
? DATA CARD
      :
      :   <user's data>
      :
? COPY YACASTEXT, USERFILM TO A111
```

If one wishes to compile a program to be saved and run later, appropriate WFL statements would be:

```
? IF FILE YACASTEXT ISNT PRESENT THEN
        COPY YACASTEXT FROM A111
? COMPILE USEREXAMPLE1 ALGOL LIBRARY
? DATA
BEGIN
 $ INCLUDE YACASTEXT
      :
      :
      :   <user program>
      :
END.
? RUN EXAMPLE1
? FILE FILM (TITLE = USERFILM)
? DATA CARD
      :
      :
      :   <user's data>
      :
      :
? COPY YACASTEXT, EXAMPLE1, USERFILM TO A111
```

No other WFL statements are required to support the use paper plotter. Files of plotter information are created and plotted automatically.

In the second example, if the program will never attempt to create any paper plots, the user may set the user option NOPAPER. This will remove some of the YACAS system routines that perform plotting and eliminate the AUTOBIND step that links the plotter routines to the user's program.

```
? COMPILE EXAMPLE2 ALGOL
? FILE FILM (TITLE = FILM2)
? DATA
 $ SET NOPAPER
BEGIN
 $ INCLUDE YACASTEXT
 ENVIRONMENT ("FILM");
      :
      :
      :
      :
END.
```

Appendix 2 Transferring Film Files

This appendix describes a technique that can be used to copy film files from the B6700 to the GT44. Users are expected to have a basic understanding of RT-11 (how to use it, how files are named, etc) and a "licence" to use the equipment. As with Appendix A, the specific details contained herein are accurate at the date of writing but may have subsequently changed. The user is advised to confirm the correctness of this information before using it.

Copying Files

The only software available to copy files from the B6700 to the GT44 is a pair of programs intended to copy paper plotter files. One of the programs runs on the B6700 as a sub-task of ODDJOB; its job is to transmit the files. The other is a stand-alone program run on the GT44 which receives the information and stores it in a pre-allocated disk file. A third program which runs under RT-11 should be used to pre-allocate the disk file (the created file is filled with Null characters). The process may be initiated and carried out entirely from the GT44.

(1) Ensure that the file to be copied is available as a B6700 disk file. This may be done by copying the file to disk just prior to the GT44 booking and hoping that it will not be deleted. Alternatively, the file may be saved on CANDEPACK and obtained from there.

(2) Ensure that the B6700 and GT44 are linked together. There is a switch in the B6700 machine room that must be set; the operators may be asked to ensure that it is. Also, in the Graphics Workroom, there is a line-driver connected to the GT44; this must be ON (it usually is).

(3) Load an appropriate disk as the system disk for the GT44

and boot RT11. PLABAK and the other programs to be described are all available on disk COSC05.

(4) Pre-allocate a disk file to hold the transmitted film file. Use the program MAKPLT as indicated below. The value in the square brackets is the size of the file to be allocated (a decimal number). YACAS produces an estimate of the size of that will be needed; it is a good idea to pre-allocate a few more blocks than this estimate as a safety margin.

```
.R MAKPLT
*FILM1.BAK[200]=
FILE COMPLETE
```

(5) Run the program GTSYS which communicates with the B6700 and loads the PDP-11 disk file with information transmitted to it. Note that once the program has started, the user must use the Burroughs line terminator (a line-feed character) rather than the PDP-11 one (a Return character). The symbols "<LF>" and "<CR>" will be used to denote these characters.

```
.R GTSYS          <CR>
ENTER DISK WORK FILE NAME
^B
^N
FILM1.BAK          <LF>
   0 PLOTS FOUND
```

Note: (a) The ^B ensures that the line to the B6700 is clear and working. Its use is optional.

(b) The ^N informs GTSYS that the file is not to be plotted, just loaded on the disk. Its use is essential!

(6) Have ODDJOB begin sending the file from the B6700 (the "!" represents the Control-F character):

```
!PLOT (COSC023)FILM1  <LF>
TRANSMITTING
RECEIVED
RECEIVED
RECEIVED
    :
    :
TRANSMITTED
^E
END OF PLOT 0, 0 CARDS, 1 BLOCKS
```

Note: (a) The complete B6700 filename is needed by the PLOT command.

(b) The ^E is necessary (wait until "TRANSMITTED" has been printed). It causes the last partial block of the file to be written to the disk.

(7) Return to RT-11 (GTSYS is a stand-alone program - go back to using the Return key as the line delimiter).

```
^C
RT-11SJ V02C-02C
.DATE 1-FEB-78 <CR>
```

(8) The file that has been copied now needs to be re-formatted for use by PLABAK. The program TBSCAN will do this:

```
.R TBSCAN     <CR>
*** FILM SCAN ***
*FILM1.FLM=FILM1.BAK/C  <CR>
** CRUNCH **
*^C
.
```

Note: The "/C" is necessary. Without it, TBSCAN will treat the input file as a film file and produce an interpreted listing of the file's contents.

A complete example of this procedure is shown below. Note that in this example the file to be transferred has been saved in the user's online disk area on CANDEFILES.

```
.DATE 1-FEB-78     <CR>
.R MAKPLT          <CR>
*FILM1.BAK[100]=   <CR>
FILE COMPLETE

.R GTSYS           <CR>
ENTER DISK WORK FILE NAME
^B
^N
FILM1.BAK          <LF>
  0 PLOTS FOUND

! PLOT (COSCO23)FILM1 ON CANDEFILES  <LF>
TRANSMITTING
RECEIVED
RECEIVED
   :
   :
TRANSMITTED
^E
END OF PLOT 0, 0 CARDS, 1 BLOCKS
^C

RT-11SJ     V02C-02C
.DATE 1-FEB-78      <CR>
```

```
.R TBSCAN          <CR>
*** FILMSCAN ***
*FILM1.FLM=FILM1.BAK/C  <CR>
** CRUNCH **
^C
.
```

To display the transferred film file, run the PLABAK program as described in section 4.

```
.R PLABAK        <CR>
** PLABAK **
WHICH FILM (FILE) * FILM1  <CR>
*** BEGIN ***  <CR>
*
```

Appendix 3 FILMMAKER

In order to allow films to be made quickly and cheaply, a simple "language" has been implemented. The language consists of most of the commands of YACAS with no variables, arithmetic statements, or program-flow control statements. The language is interpreted by a program called FILMMAKER (available on tape B172).

Syntax

(1) The first card is an initialization card. It contains four values separated by commas.

(a) The first value is the ENVIRONMENT command parameter. Allowed values are "FILM", "PAPER", "BOTH" or "NONE". One of these values is necessary.

(b) The second value is the UNITS command parameter. Allowed values are "FRAMES", "SECS", or a positive whole number > 0.

(c) The third and fourth values are the parameters for the CARTOONING command (see example below).

(2) Subsequent cards may be blank, comment cards, or command cards. Each card must be a complete blank, comment, or command; continuation cards are not allowed. Only the command cards are interpreted, and these are interpreted in a strictly sequential order. Commands may appear in any order within the restrictions imposed by YACAS.

(3) A comment card is one on which the first character is an asterisk ("*").

```
eg:  * This is a comment card
          * And so is this
```

(4) A command card consists of the command name (only the first six characters are necessary) followed by the parameters required by that command, separated by commas. The parameters must be numeric values or literal strings of

1 to 6 characters. The command parameters may be followed by a comment provided an asterisk separates the parameters and comments.

```
eg: OPENP,1,      *THIS IS A COMMENT
    ARC,2,2,2,12,0,7,1,
    CLOSEP,
    UNITS,"SECS",     *THIS IS ANOTHER COMMENT
    INTERVAL,0,5,
    MOVETO,1,10,7.5,
```

(5) Pictures are identified by a user-specified number in the range 1 to 50. The OPENP command is slightly different than that described for YACAS in that the user-specified picture number is given as a parameter to the command (see example above).

(6) Parameter values may be given in any single-precision form recognized by ALGOL. Input formats I, E, F, and R in particular may be used, and literal strings of 1 to 6 characters (where acceptable by YACAS) may be also used. (FILMMAKER uses the Burroughs ALGOL "free-field" input format to read the command parameters. See pages 5-64 and 5-65 of the ALGOL manual for details).

FILMMAKER Commands

FILMMAKER recognizes most of the YACAS commands listed in section 3 of the User Guide. Those that it does not recognize are listed below, and are marked with an asterisk ("*") in section 3.

```
CAPTION
DUMPSTATE
ENVIRONMENT (implicity recognized via card 1)
FINISHED
```

FILMMAKER also recognizes the commands BOX, SHEAR and ACCELERATE as described in section 5.3.3. It does not recognize any of the supplementary commands listed in section 5.5.

Using FILMMAKER - An Example.

FILMMAKER is available as an executable program on tape B172 (account COSC023). The input file for FILMMAKER is called CARD and is declared to have KIND=READER. Appropriate WFL statements to use FILMMAKER are shown in the following example:

```
? COPY (COSC023)FILMMAKER AS FILMMAKER FROM B172
? RUN FILMMAKER
? DATA CARD
 "PAPER","FRAMES",4,1,     * SETUP
 OPENP, 1,                 * PIC 1
 BOX,10,7.5,5,3,           *     A BOX
 CLOSEP
 DRAWF, 1,                 * DRAW IT
 INTERVAL,0,6,             * FOR 6 FRAMES
 SHEAR,1,30,               *     SHEAR IT
 CUSHIONED,
 ROLLIT,"*","*",
 DRAWF, 1,                 * DRAW IT SHEARED
            ** T h a t ' s   A l l **
? END JOB
```

FILMMAKER prints each card it reads from the input file on the YACAS log file (LP) interspersed with YACAS messages.

Appendix 4 READCEL Data Card Format

Data cards for the READCEL command may be prepared in a free format. Blanks are significant: they act as delimiters for numbers and so must not be embedded in the numbers; one or more consecutive blanks are treated as a single blank. Card boundaries are generally treated as a single blank except as noted below. Cards may contain comments. A comment begins with a "%" symbol and ends with either another "%" or a card boundary. A comment is treated, syntactically, as a blank.

A READCEL picture description consists of three groups of (X,Y,C) values, one for each of the cel's detail, outline, and attachment point sections. X,Y is a point's coordinates, and C is its connectivity (as used in the VECT command). Each group is terminated with a "#" symbol. Cel descriptions may also contain a name for the cel (a character string of up to 24 characters enclosed in '"' or "'" symbols such as "NAME" or 'THIS NAME'). It may also have a pivot point (two numbers representing the X,Y position of the pivot point, enclosed in "<" and ">" brackets: <15,5>). The cel name and pivot point are optional and if provided may appear anywhere in the cel description prior to the "#" terminating the attachment point section.

Each cel description must begin on a new card.

The numbers used for points may have any of the ALGOL I, F, E, or R forms. Numbers may be signed with the characters "+" or "-". They should be separated from each other by blanks or "," characters, though the sign characters will do where their use is unambiguous. Valid connectivity codes are the numbers 1 and 0 as described for the VECT command. Alternatively, the special symbols ":"

and ";" may be used for 1 and 0 respectively.

Note: any characters not in the following set are treated as blanks:

0123456789"'():;#+-.@E

The following examples will serve to illustrate these points.

Example 1:

A cel consisting of a rectangular box of detail vectors only. No name or pivot point is specified.

```
0,0,1 5,0,0 5,3,0 0,3,0 0,0,0 ###
```

Example 2:

A cel consisting of two triangles in the detail section and a square outline. A name and pivot point are specified. Note the use of the alternate connectivity code characters ";" and ":".

```
"Pic 1" <4,4> 2,2: 4,2; 3,4; 2,2;
5,5: 6,7; 4,7; 5,5; # 0,0: 10,0; 10,10; 0,10; 0,0; ##
```

Example 3:

A cel consisting of a triangle in the detail section, a square outline, and one attachment point. A pivot point is specified, but no name.

```
<3,5>    %The pivot point is at <3,5>%
         %The detail section%
4.25,3.50,1  3.15,2.0,0  5.35,2.0,0  4.25,3.50,0  #
         %Now the outline%
-10,-5,1  1.0E1,-5.0,0  1.0@+1,500E-2,0
-1.0E1,+0.5@1,0  -10,-5,0  #
         %Now the attachment point%
-4.0,+3.0,1   #
```

Appendix B

An Example

The program provided as an example of the use of YACAS, Can be used to make films showing the operation of an "exchange sort". The program is given a set of from 2 to 10 numbers (with magnitudes in the range 3 to 10 - duplicates are allowed). These will be sorted into either ascending or descending order.

The film produced by the program is similar to that shown in the paper plots below. The numeric values are represented by rectangles whose X dimension is proportional to the magnitude of the value. Arrows are used to show which elements of the set are being inspected. The arrow on the left points to the "bottom" of the set; it is moved upwards as the lower positions are filled in with their correct values. The arrow on the immediate right points to the value to be exchanged for the bottom one (the largest of those remaining if the sort is to be descending, the smallest if the sort is ascending). The right-most arrow points to the element of the set being examined for relative magnitude.

The program is "controlled" with three cards. The first contains four values to be used directly in the ENVIRONMENT, UNITS, and VIEWPORT commands of YACAS. The second card contains 8 values used to control the timing of the film. The third contains the set of values to be sorted and a code for the direction of the sort. All values are read using the Burroughs ALGOL "field free" format.

E X A M P L E

```
BEGIN                                   % EXAMPLE PROGRAM 1                                    B.0000 IS
$ INCLUDE YACASTEXT                                                                          1
  INTEGER  BIGMARK, SMALLMARK, BOTTOMP;
  INTEGER  I, J, K, N, T;
  INTEGER ARRAY  P[1:10], TABLE[1:10];
  REAL     INP, OUTP, INB, OUTB, PAUSE1, PAUSE2, SHIFT, COMPAR;
  REAL     A,B,R,X,TIME;
  REAL ARRAY  PY[1:10];
  BOOLEAN  DOWNSW;

PROCEDURE  CREATEPICS;
COMMENT
   CREATE THE PICTURES FOR THE FILM;
    ITEMS APPEAR AS BOXES WHOSE LENGTH IS PROPORTIONAL TO THE ITEM VALU
    POINTERS APPEAR AS ARROWS;
BEGIN  INTEGER I;                                                                  CREATEPICS IS
                                        % PICTURES OF ITEMS TO BE SORTED            2
  R := .5 + 5./N;                 % (HEIGHT OF BOXES)
  A := .5 + R*.5;                 % (STARTING POSITION)
  FOR I := 1 STEP 1 UNTIL N  DO BEGIN
    X := .5+.5*TABLE[I];     % (LENGTH OF BOX)                                       3
    PY[I] := A;    A := A + R*1.4;      % POSITION OF THIS BOX
    P[I] := OPENP;
      VECT(1,0.,0.); VECT(0,X,0.);  VECT(0,X,R);
      VECT(0,0.,R);  VECT(0,0.,0.);
      PIVOT( 2.5, R*.5 );  BRIGHTNESS( 0. );
      CLOSEP;
    MOVETO( P[I], 10., PY[I] );  % PUT BOX AT START POSITION
     END;
                                % MARKERS                                            3
  SMALLMARK := OPENP;
    VECT(1,0.,0.);  VECT(0,.4,.4);  VECT(0,.4,.2);  VECT(0,1.2,.2);
    VECT(0,1.2,-.2);  VECT(0,.4,-.2);  VECT(0,.4,-.4); VECT(0,0.,0.);
    PIVOT( -4., 0.); BRIGHTNESS( 0. );
    CLOSEP;
  BOTTOMP := OPENP;                 % BOTTOM MARKER
    USECEL( SMALLMARK, 0., 0., 0., 0., 180.);
    BRIGHTNESS( 0. );
    CLOSEP;
  BIGMARK := OPENP;
    VECT(1,0,0);  VECT(0,.4,.5);  VECT(0,.4,.3);  VECT(0,1.2,.3);
    VECT(0,1.2,-.3);  VECT(0,.4,-.3);  VECT(0,.4,-.5);  VECT(0,0,0);
    PIVOT( -6.0, 0.);  BRIGHTNESS( 0. );
    CLOSEP;
END  CREATEPICS;                                                                   CREATEPICS(068
                                                                                     2
PROCEDURE STARTFILM;
COMMENT
   PERFORM THE OPENING SEQUENCE FOR THE FILM;
BEGIN  INTEGER I;
                                                                                   STARTFILM IS
  TIME := 0;                                                                       2
                                    % FADE IN THE PICTURES TO BE SORTED
                                    % FROM BOTTOM TO TOP.
  FOR I := 1 STEP 1 UNTIL N  DO BEGIN
    INTERVAL(TIME+1, INP );  TIME := TIME + INP;                                   3
    FADE( P[I], 5.);
    END;
                                    % POSITION THE  BOTTOMP AND                    3
                                    % SMALLMARK  PICTURES, THEN FADE THEM
  SETSTATIC;
  MOVETO(BOTTOMP, 10., PY[1] );
  MOVETO( SMALLMARK, 10., PY[1] );
  INTERVAL( TIME+1, INP*2 );
    FADE( BOTTOMP, 4.);
```

```
    INTERVAL( TIME+INP, INP*2 );
      FADE( SMALLMARK, 4.);
      TIME := TIME+INP*2;
                                % NOW ROLLIT FOR THIS SEQUENCE, ADDING
                                % A PAUSE.
    INTERVAL( 0., TIME+PAUSE1 );
      FOR I := 1 STEP 1 UNTIL N  DO  DRAW( P[I] );
      DRAW( BOTTOMP );  DRAW( SMALLMARK );
    ROLLIT( 0., TIME+PAUSE1 );
    TIME := 0.;
  END STARTFILM;
                                                                    STARTFILM(069
                                                                    2
  PROCEDURE STARTSTEP( I );
    VALUE I;  INTEGER I;
  COMMENT
    SHIFT, FADE IN, ETC.  PICTURES TO GET READY FOR STEP I;
  BEGIN
    IF I > 1  THEN BEGIN        % UNLESS THIS IS THE FIRST STEP
                                % MOVE BOTTOMP AND SMALLMARK UP ONE          3
                                % ELEMENT.
      INTERVAL( TIME+1, SHIFT );  TIME := TIME + SHIFT;
      MOVETO( BOTTOMP, 10., PY[I] );  CUSHIONED;
      MOVETO( SMALLMARK, 10., PY[I] );  CUSHIONED;
      END;
                                % MOVE THE BIGMARK TO POSITION I AND          3
                                % FADE IT IN.
    SETSTATIC;
    MOVETO( BIGMARK, 10., PY[I] );
    INTERVAL( TIME+1, INB );
      FADE( BIGMARK, 4.);
                                % ADD A SHORT PAUSE BEFORE BEGINNING.
    TIME := TIME+INB+PAUSE1;
  END STARTSTEP;
                                                                    2
  PROCEDURE SHIFTMARK( MARK, J );
    VALUE MARK, J;  INTEGER MARK, J;
  COMMENT
    MOVE PICTURE 'MARK' TO POSITION J;
  BEGIN
    INTERVAL( TIME+1, SHIFT );  TIME := TIME + SHIFT;
      MOVETO( MARK, 10,  PY[J] );  CUSHIONED;                           2
  END SHIFTMARK;
                                                                    2
  PROCEDURE COMPAREPICS( J, K );
    VALUE J, K;  INTEGER J, K;
  COMMENT
    SEQUENCE TO INDICATE WHICH PICTURES ARE BEING COMPARED;
  BEGIN
                                % SHOW THE COMPARISON BY ENLARGING THE
                                % PICTURES, THEN SHRINKING THEM TO NORMAL.
    INTERVAL( TIME+1, COMPAR);  TIME := TIME+ COMPAR;
      SCALEBY( P[J], 1.3, 1.3 );  SINUSOID( 1. );                       2
      SCALEBY( P[K], 1.3, 1.3 );  SINUSOID( 1. );
  END COMPAREPICS;
                                                                    2
  PROCEDURE    SWAPITEMS( I, K );
    VALUE I, K;  INTEGER I, K;
  COMMENT
    IF I AND K DIFFER, SWAP THE POSITIONS OF THE ITEMS (AND SWAP THEIR
    PICTURE NUMBERS).  IF I = K, SIMPLY PAUSE;
  BEGIN  INTEGER M;
                                                                    SWAPITEMS IS
    IF I NEQ K  THEN BEGIN                                          2
                                % IF I AND K DIFFER, SWAP THE PICTURES AND   3
                                % THE SMALLMARK (I ASSUMED TO BE THE LOWER
                                % POSITION).
      INTERVAL( TIME+1, SHIFT );  TIME := TIME + SHIFT;
      MOVETO( P[I], 10., PY[K] );  CUSHIONED;
      MOVETO( P[K], 10., PY[I] );  CUSHIONED;
      MOVETO( SMALLMARK, 10., PY[I] );  CUSHIONED;
      M := P[K];  P[K] := P[I];  P[I] := M;
      END
    ELSE
      TIME := TIME + PAUSE1;                                        3
  END SWAPITEMS;
                                                                    SWAPITEMS(06A
```

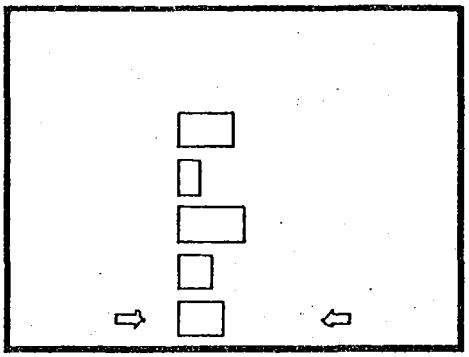
```
                                                                        2
PROCEDURE ENDSTEP;
COMMENT
  DO A ROLLIT FOR THE THINGS STORED IN THE AGENDA;
BEGIN
  INTEGER I;                                                                ENDSTEP IS
                                % FIRST, FADE OUT THE BIGMARK.                2
  INTERVAL( TIME+1, OUTB );  TIME := TIME + OUTB;
    FADE( BIGMARK, 0. );
                                % NOW ROLLIT AFTER A SHORT PAUSE.
  INTERVAL( 0., TIME+PAUSE1 );
    FOR I := 1 STEP 1 UNTIL N DO  DRAW( P[I] );
    DRAW( BOTTOMP );  DRAW( SMALLMARK );  DRAW( BIGMARK );
  ROLLIT( 0., "*" );
  TIME := 0.;
END ENDSTEP;
                                                                        ENDSTEP(066)
                                                                        2
PROCEDURE ENDFILM;
COMMENT
  PERFORM THE FILM CLOSING SEQUENCE;
BEGIN  INTEGER I;                                                         ENDFILM IS
                                % FADE OUT THE SMALLMARK (THE BIGMARK IS      2
                                % ALREADY OUT), SIMULTANEOUSLY MOVING
                                % BOTTOMP TO THE TOP POSITION.
  INTERVAL( TIME+1, SHIFT );  TIME := TIME + SHIFT;
    FADE( SMALLMARK, 0.);
    MOVETO( BOTTOMP, 10., PY[N] );  CUSHIONED;
                                % A SHORT PAUSE, THEN SCAN DOWN,
                                % FADING OUT ON REACHING THE BOTTOM.
  TIME := TIME + PAUSE1;
  INTERVAL( TIME+1, SHIFT*3 );  TIME := TIME + SHIFT*3;
    MOVETO( BOTTOMP, 10., PY[1] );  CUSHIONED;
  INTERVAL( TIME+OUTP, OUTP );
    FADE( BOTTOMP, 0.);
                                % SHORT PAUSE, THEN FADE OUT THE ITEMS ONE-
                                % BY-ONE FROM THE TOP.
  TIME := TIME + PAUSE1;
  FOR I := N STEP -1 UNTIL 1 DO BEGIN
    INTERVAL( TIME+1, OUTP );  TIME := TIME + OUTP;                         3
    FADE(-P[I] , 0.);
    END;
                                % NOW ROLLIT ALL.                           3
  INTERVAL( 0., TIME );
    FOR I := 1 STEP 1 UNTIL N DO  DRAW( P[I] );
    DRAW( BOTTOMP );  DRAW( SMALLMARK );
  ROLLIT( 0., "*" );
END ENDFILM;
                                                                        ENDFILM(066)
                                                                        2

        %*****************************
        %***  MAIN PROGRAM CODE  *****
        %*****************************

READ(CARD,//, A,B,R,X );
WRITE(LP,<"FILMING ENVIRONMENT: ",C6," - TIME UNITS: ",C6>, A,B);
READ(CARD,//, INP,OUTP,INB,OUTB,PAUSE1,PAUSE2,SHIFT,COMPAR );
WRITE(LP,<"TIMING VALUES: ",8F6.1>,
                        INP,OUTP,INB,OUTB,PAUSE1,PAUSE2,SHIFT,COMPAR );
ENVIRONMENT(A);  UNITS(B);
                                                                          DATA
IF (A = "PAPER")  OR  (A = "BOTH")  THEN BEGIN                          2
    IF (R < 1)  OR (R > 4)  THEN R := 4;                                2
    IF (X < 4) THEN  X := 24;
    CARTOONING( R, X );
    WRITE(LP,<"     CARTOONING MODE: ",I3,",",I6>, R,X );
    END;
READ(CARD,//, N, FOR I := 1 STEP 1 UNTIL N DO  TABLE[I], B );           2
WRITE(LP,<"SORT",I3," VALUES: ",*(I3,","),N>, N, N,
                        FOR I := 1 STEP 1 UNTIL N DO  TABLE[I] );
WRITE(LP,<"DIRECTION: ", C6>, B );
DOWNSW := (B = "DOWN");
CREATEPICS;
STARTFILM;
```

```
                FOR I := 1 STEP 1 UNTIL (N-1)  DO BEGIN
                  STARTSTEP( I );                                              2
                  K := I;
                  FOR J := (I+1) STEP 1 UNTIL N  DO BEGIN
1                   SHIFTMARK( BIGMARK, J );                                   3
2                   COMPAREPICS( J, K );
3                   IF ( DOWNSW AND (TABLE[J] > TABLE[K]) )
                          OR ( NOT DOWNSW AND (TABLE[J] < TABLE[K]) )  THEN BEGIN
4                     SHIFTMARK( SMALLMARK, J );                               4
5                     K := J;
6                     END;
                    END;                                                       4
7                 IF K NEQ I  THEN BEGIN                                       3
8                   T := TABLE[K];  TABLE[K] := TABLE[I];  TABLE[I] := T;       3
                    END;
9                 SWAPITEMS( I, K );                                           3
                  ENDSTEP;
                  END;
                ENDFILM;                                                       2
                WRITE(LP,<"SORTED TABLE: ",*(I3,",")>, N,
                                          FOR I := 1 STEP 1 UNTIL N DO  TABLE[I] );
                FINISHED;
              END OF MAIN.
```

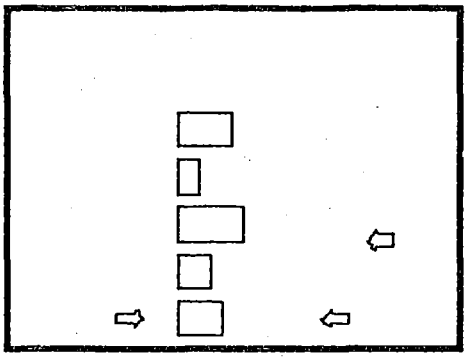
```
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 110.  TOTAL SEGMENT SIZE = 6747 WORDS.  CORE ESTIMATE = 11067 WORDS.
PROGRAM SIZE = 2674 CARDS(2 OMITTED CARDS), 20101 SYNTACTIC ITEMS, 551 DISK SEGMENTS.
PROGRAM FILE NAME:  EXAMPLE.
COMPILATION TIME = 61.946 SECONDS ELAPSED;  30.543 SECONDS PROCESSING;  10.075 SECONDS I/O.
```

```
FILMING ENVIRONMENT: ??BOTH - TIME UNITS: FRAMES
TIMING VALUES:   16.0  15.0  12.0  12.0  12.0  18.0  30.0  42.0
BOTH PAPER AND FILM SELECTED FOR OUTPUT
     CARTOONING MODE:   4,    63
SORT  5 VALUES:   3,  2,  5,  1,  4,
DIRECTION: ??DOWN
AT   1, ROLLIT FROM   1 TO 138
OPEN PLOTTER ... LENGTH =  2600
AT 139, ROLLIT FROM 139 TO 534
AT 535, ROLLIT FROM 535 TO 918
AT 919, ROLLIT FROM 919 TO1152
AT1153, ROLLIT FROM1153 TO1362
AT1363, ROLLIT FROM1363 TO1582
SORTED TABLE:   5,  4,  3,  2,  1,
FILM WILL REQUIRE   253 PDP-11 DISK BLOCKS
  1582 FILM FRAMES PLOTTED
    26 FRAMES DRAWN ON PLOTTER
*** FINISHED: CURRENT FRAME IS  1583 ***
```

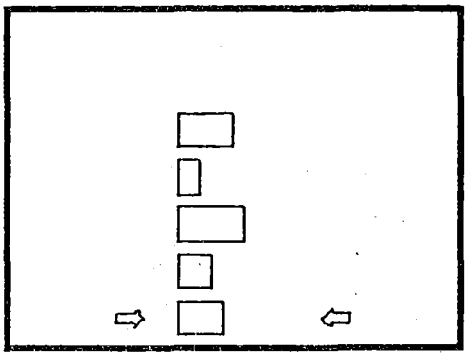
FRAME 00001



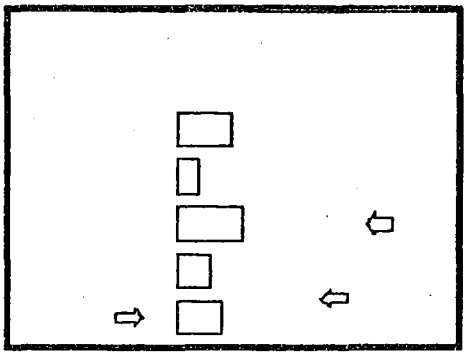
FRAME 00253



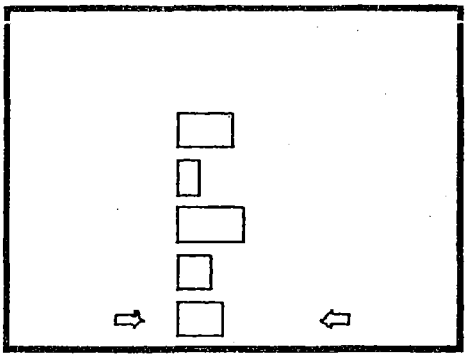
FRAME 00064



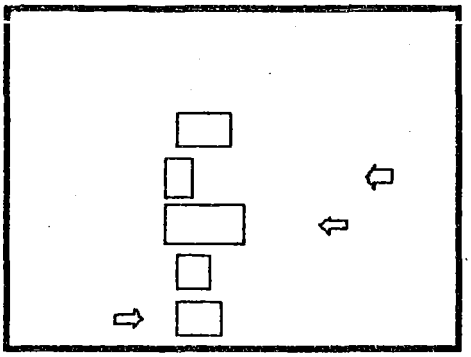
FRAME 00316



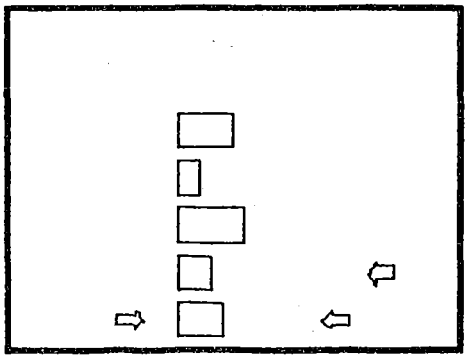
FRAME 00127



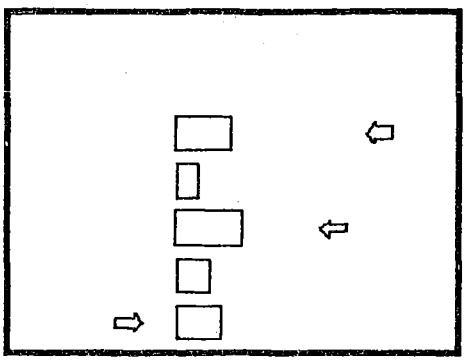
FRAME 00379



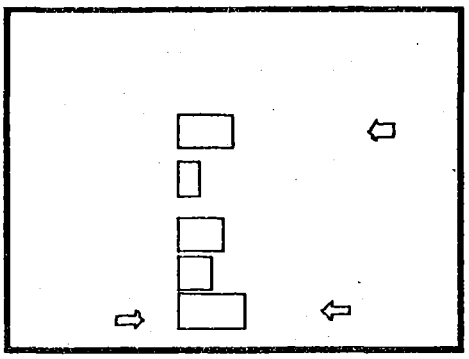
FRAME 00190



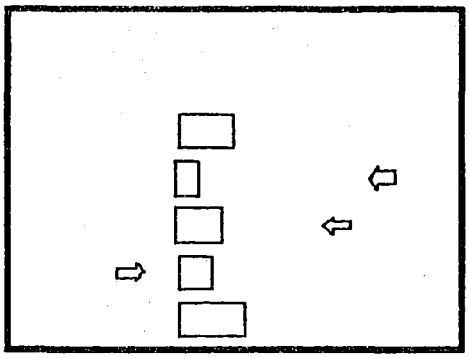
FRAME 00442



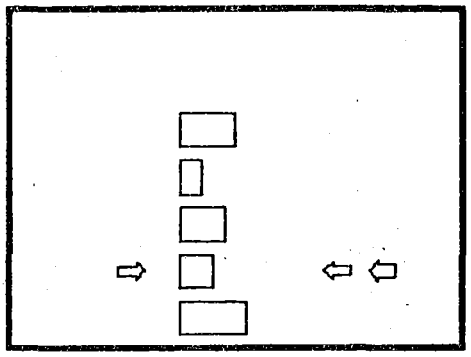
FRAME 00505



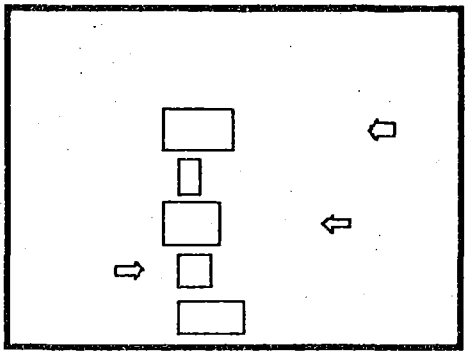
FRAME 00757



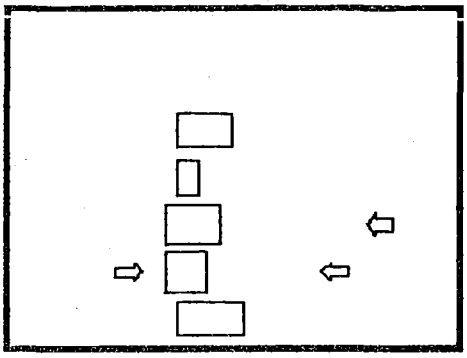
FRAME 00568



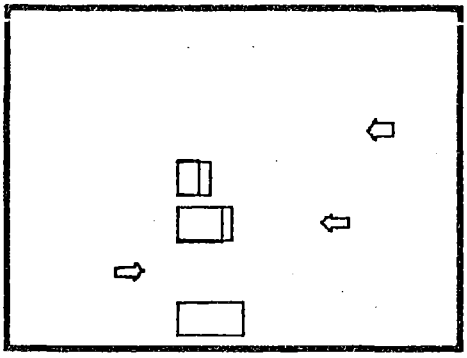
FRAME 00820



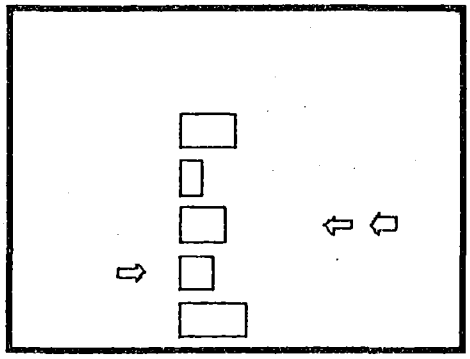
FRAME 00631



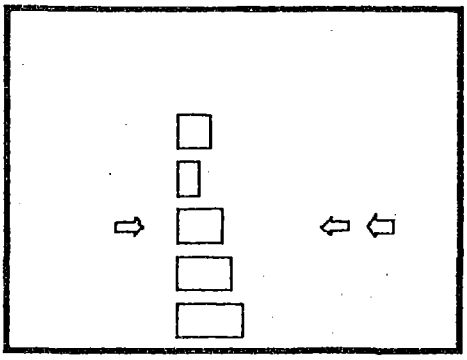
FRAME 00883



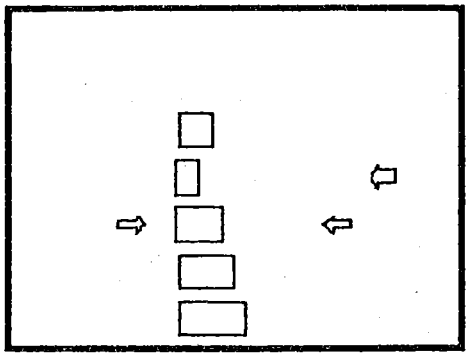
FRAME 00694



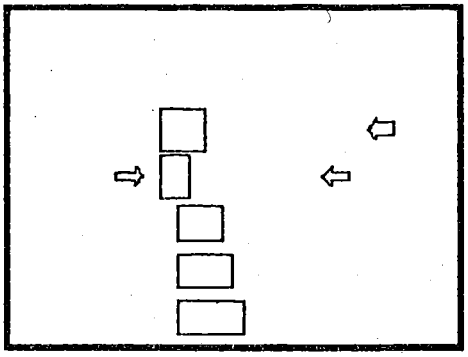
FRAME 00946



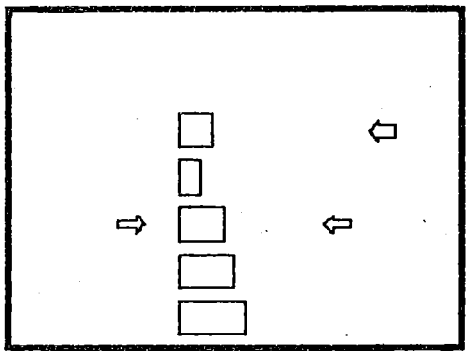
FRAME 01009



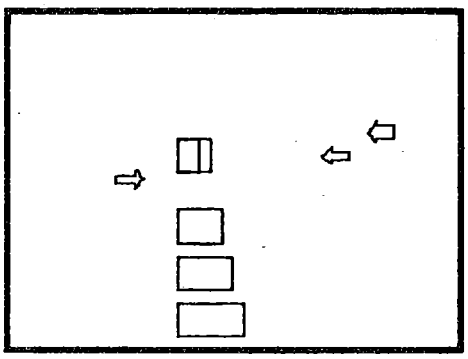
FRAME 01261



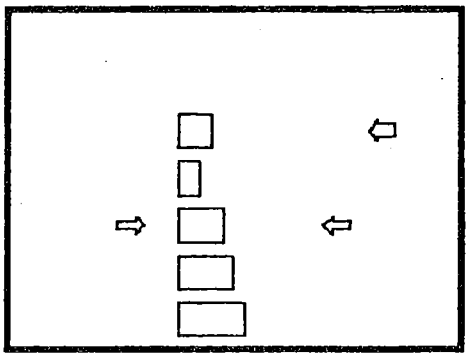
FRAME 01072



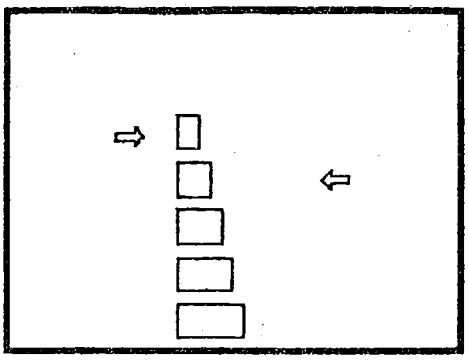
FRAME 01324



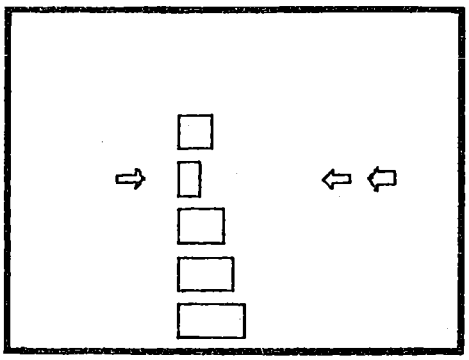
FRAME 01135



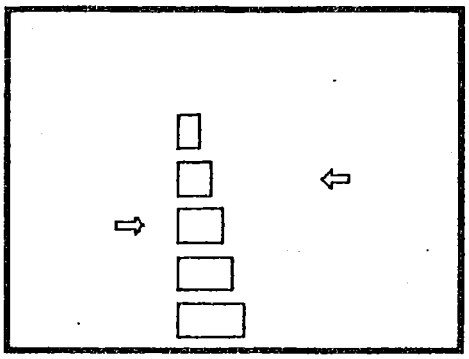
FRAME 01387



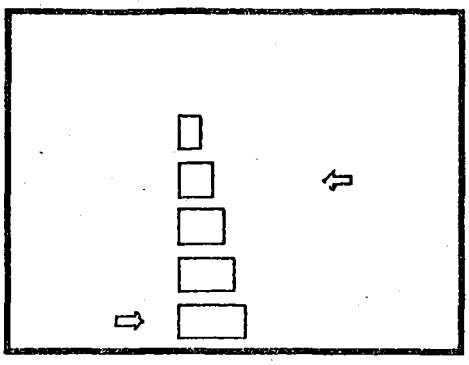
FRAME 01198



FRAME 01450



FRAME 01513



FRAME 01576